DESIGNING FOR PRIVACY IN INTERACTIVE SYSTEMS

A Dissertation
Presented to
The Academic Faculty

by

Carlos Jensen

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
College of Computing

Georgia Institute of Technology
December 2005

DESIGNING FOR PRIVACY IN INTERACTIVE SYSTEMS

Approved by:

Dr. Colin Potts, Advisor
College of Computing
Georgia Institute of Technology

Dr. Julia B. Earp
College of Management
North Carolina State University

Dr. Annie I. Antón
Department of Computer Science
North Carolina State University

Dr. W. Keth Edwards
College of Computing
Georgia Institute of Technology

Dr. Amy S. Bruckman
College of Computing
Georgia Institute of Technology

November 14, 2005

This dissertation is dedicated to the memory of my father.

Eliot J. Jensen
1942 – 1995

For always wanting me to succeed and showing me that parents are people too.

ACKNOWLEDGEMENTS

This dissertation is the culmination of many years work, work which would not have been possible without the help and support of a great many family members, friends, and colleagues.

While this dissertation is dedicated to my father, it is equally to the credit of my mother, who for all these years was a constant source of encouragement, sympathy and support. Thanks for never loosing faith, and always being there whenever I needed someone to talk to, even if about nothing.

I thank my wife for putting up with the all-nighters, deadlines, the glamorous grad-student lifestyle, my grumpiness and frustration, and for taking far better care of me than I deserved. She also deserves thanks for our wonderful daughter, the best source of distraction, stress relief, perspective, pride, and joy I have ever known.

I owe a great debt to my brother for putting up with endless questions and discussions about statistics and experimental design, and for volunteering to proof so many papers and drafts over the years. I also thank my sister for always being herself, and always putting up with her big brother.

I would like to thank my advisor, Colin Potts for being all-round wonderful and inspiring, when I grow up I hope to be just like you. Colin let me run with these ideas, crazy and unstructured as they were at times, yet was always there to say just the right thing, or ask just the right questions to move things in the right direction. He has always been in my corner, and where it not for all the help and encouragement during these last months, I don’t think I could have done this. Thank you.

I also need to thank my committee, Annie I. Antón, Julia B. Earp, and W. Keith Edwards for all their comments and feedback on the direction of this work as well as drafts of this document. Special thanks go to Amy S. Bruckman, who in addition to being on the committee also supported and guided me through my first years in grad school, and offered me some wonderful opportunities for outside collaboration and networking.

I would like to thank my friends and colleagues, Joe Tullio and Khai Truong, who have helped me refine this work, helped me with experiments, and read over early drafts of this and other documents. They have been great friends through these years, and great collaborators. Along these lines I'd also like to thank Heather Richter for letting me use part of her work in my experiments. I also need to thank Carl Cox and Chris Morris my team-mates in the requirements engineering class which led to this dissertation topic.

I also need to thank the undergraduates who have worked with me on this project over the years, Ji Han Bae, John O. Ndukuba, Robert Marinski, and Leandro Taberner, who all managed to graduate before me. Thank you all for your wonderful work and the opportunity to learn a little about mentoring. I wish you the best of luck wherever you may be, and wherever life may take you.

Though there are many faculty members whom I should thank for their help and guidance over the years, I would especially like to mention Norberto Ezquerra for involving me in his study abroad program, not only giving me an opportunity to learn to teach, but also giving my family a wonderful opportunity to get to know and experience Barcelona.

I also want to thank those outside of Georgia Tech who have given me opportunities and taught me a great deal. In particular Shelly Farnham and John Davis, my mentors during my internships at Microsoft Research who taught me most of what I

know about experimental design and analysis, and who taught me to appreciate psychologists. Similarly, I would like to thank the people of the Privacy Place, past and present who have been part of this project.

I thank my other friends and colleagues, many of whom helped me directly in some aspect of this work, and many other who were simply there to make the experience more enjoyable. I'd like to thank the original ELC crew, Jason Elliott, Jason Ellis, Josh Berman, Lizzie Edwards, and Rodney Walker for making grad-life fun and interesting. I'd also like to thank the extended GVU grad-student family, including Jessica Elliott, Elaine Huang, Idris Hsi, Michael Terry, Jeff Rick, Jim Hudson, and many, many others. Thanks for making grad-school a fun and enjoyable place to be.

I would also like to thank the staff of the GVU and the College of Computing who have been providing help and support every step of the way. You have been absolutely wonderful.

Finally, I thank Kulathur Rajasethupathy, Raymond Duncan, and Peter Marchant, my mentors at SUNY Brockport, all-round wonderful and inspirational people who each in his own way changed my life.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

SUMMARY

Current models for privacy-aware design were examined and compared to priorities and needs of end-users as determined from a number of studies presented. Based on these studies, we predicted these frameworks to be sub-optimal because of either a lack of structure in the analysis task, or too high a cost. To examine this point a new design framework combining the advantages of previous frameworks with a light-weight goal-oriented analysis technique. This new framework, STRAP (Structured Analysis of Privacy), was predicted to out-perform existing frameworks in terms of effectiveness (overall detection of privacy issues) and efficiency (number of privacy issues discovered over time on task or number of independent analysts).

Three design experiments were designed to study the relative effectiveness and efficiency of these design frameworks. A total of eighty-five subjects took part, analyzing systems from three different application domains in order to demonstrate the flexibility and adaptability of these frameworks. These experiments confirmed that though existing frameworks were generally effective they were not generally efficient. STRAP on the other hand was shown to be both efficient and effective, validating our earlier analysis. Overall, these findings show that privacy can effectively and efficiently be considered as part of the system design process. Further findings and implications as well as limitations and future work are discussed.

CHAPTER 1
INTRODUCTION

People are increasingly concerned about their online privacy and how computers are used to collect, process, share, and store personal information, as expressed in numerous surveys (Culnan 1999, Earp & Meyer 2000, Kobsa 2002). These concerns are a natural consequence of the growing number of privacy invasions (FTC 2000, Synovate 2003, Javelin 2005), the pervasiveness of information capture and sharing in IT systems, and an increasing public awareness of these problems. These developments have led to legislative efforts to protect an individuals' privacy, limiting what systems and data collectors may do, and what safeguards they must offer the data subject (such as the EU Privacy Directive (EU, 1995), the US Health Insurance Portability and Accountability Act of 1996 (HIPAA) (US 1996), the US Gramm-Leach-Bliley Act of 1999 (GLBA) (US 1999), the US Children's Online Privacy Protection Act of 1998 (COPPA) (US 1998)). Privacy is an especially important concern in the areas of computer supported collaborative work (CSCW), e-commerce, and other online systems, which often require some information collection and sharing. These systems depend on the trust and confidence of the user, who must choose to participate and share information. As Grudin (1994) showed, privacy issues, real or imagined, can seriously affect the adoption and success of these systems. In order to minimize risks to users, avoid legal liabilities, and minimize maintenance costs, it is important to identify and address potential privacy issues before the development and deployment of systems.

As a design problem, privacy is difficult to address because it forces the system designer to tackle many difficult questions at once. Legal requirements must be

determined and met, and if possible, the design must stand up to scrutiny on this count. This has become increasingly important as the number of laws and the financial penalties for violations increase. System designers must increasingly deal with a complex legal landscape, especially in systems spanning borders and legal jurisdictions.

An increasingly common requirement in new legislation is for users to make informed decisions about the collection, use and dissemination of their personal information, and for systems to support this process. In order to do this, system designers must understand and accommodate user requirements and preferences when it comes to privacy, as well as provide adequate support for privacy management, which includes providing awareness, decision support, and adequate controls. Even when not subject to legal requirements, this is often desirable, as it helps build trust in the system.

The role of usability in privacy and security management has been recognized as a major challenge to widespread adoption of such systems (CRA, 2003). Whitten and Tygar (1999) found that in the context of security, well-engineered systems often fail or are intentionally circumvented because of inadequate usability. This goes beyond the initial problems associated with installation and configuration of such systems. Even when interfaces are well-designed, most people do not actively manage their security (Weirich & Sasse, 2001).

The lack of interest in privacy management, despite wide-spread concern about privacy invasions, has several underlying reasons. While users are constantly reminded of the inconveniences associated with using such systems, they are seldom made aware of the risks they have avoided through their use. As a consequence, there exists a perception-gap; most users do not see themselves as potential targets or victims and are therefore likely to disable and circumvent security measures when they judge them to be

too much of a hindrance. This has important implications for designers who must seek to strike a balance between engaging users and minimizing the burden and distraction to them as they go about their task.

It is important to note that although privacy and security are closely related, they are fundamentally different problems. In security, the primary concerned is the integrity of the systems; preventing outsiders from gaining access, and ensuring insiders stick to their defined roles. Another way of thinking about this is that security is about enforcing rules. In privacy, the primary concern is limiting collection, access and use of information, except where permitted by the data subject and relevant legislation. In other words, privacy can be thought of as being about making rules and agreements about the use of data.

Security is often a prerequisite for privacy, if a systems' integrity is compromised, the rules governing the use of data cannot be enforced. Likewise, the most common consequence of a security breach is a violation of privacy. It is however important to note that problems stemming from the violation of security measures, though important, are not the primary source of privacy violations. As Adams and Sasse (2001) explain, *"[m]ost invasions of privacy are not intentional but due to designers' inability to anticipate how this data could be used, by whom, and how this might affect users."* These are the types of privacy problems which are the primary focus of this thesis.

As with security, many non-trivial privacy problems are caused by high-level architectural decisions. These decisions can affect many different aspects of a system, making it difficult to gauge their overall impact. An example of such a decision could be for a system to be built on a distributed platform rather than a more classic client-server model. While reasonable from a resource and efficiency standpoint, a distributed system

may be more reliable and have higher availability, such a decision can be potentially problematic from a privacy perspective. Such a decision makes it necessary to consider where these servers will be and what legislation they are governed by, what information will be stored where, and what safeguards the system can offer against local government or other local seizure, and what security measures will be put in place for the communication between the nodes in the network. These types of issues can be hard to identify before a system is built, and both difficult and costly to address after the fact (Anderson 2001, Boehm 1981).

Given the importance of this problem it is not surprising that several design and analysis methods have been proposed to deal with these types of issues, including those by Bellotti & Sellen (1993), Bellotti (1997), Hong et al. (2004), Yu & Cysneiros (2002), Patrick & Kenny (2003), and Langheinrich (2001). In the area of privacy, the most popular approach has been to use heuristics to guide the analysis and design process. Heuristics in this context are rules of thumb, guidelines, or best lessons derived from real-world experience or sets of requirements. This use of heuristics builds on the work of Nielsen & Molich (1990), which showed that heuristics, when used by a team of analysts and with an appropriate scaffold, are a cost-effective and efficient analysis method. As part of this dissertation I will examine whether this model is appropriate for the more ill-defined problem of identifying privacy vulnerabilities.

In this dissertation I will present studies looking at people's preferences with regards to privacy, how people reason about privacy, and how current privacy management models match their needs and mental models. From these studies, as well as surveys of the literature, I derive a set of challenges for future privacy management interfaces. I also present an examination of existing analysis frameworks, looking at both

theoretical foundations as well as the results of a set of design experiments designed to examine their relative merits and strengths. In this context, I introduce a light-weight structured analysis framework (STRAP) which addresses some of the theoretical shortcomings identified in existing frameworks, and compare it to existing frameworks.

1.1 Thesis Statement

Designing for privacy is a difficult problem challenge because of the variability found in terms of what are considered acceptable privacy practices and sensitive information between different users and situation, and because relatively simple architectural decisions can have wide-ranging effects on a proposed system. A number of analysis and design methods have been proposed, though none have been extensively evaluated, and most limit their scope to the design and analysis of ubiquitous systems. In this dissertation I show that no design and analysis method is both efficient and effective for a broad class of systems, such as ubiquitous, groupware, and e-commerce systems. Through the introduction and evaluation of STRAP, I demonstrate that by building frameworks on a careful study of theoretical foundations, common problems, and users' preferences and behavior, some of these shortcomings can be overcome.

More specifically my hypothesis is that when compared to methods such as Bellotti & Sellen (1993), Hong et al. (2004), and Patrick and Kenny (2003), STRAP is:

1. Applicable to a broader class of systems

   The majority of frameworks limit their claims about their usefulness to the study and design of ubiquitous systems. I will show that STRAP is a useful method for a larger class of systems, that of interactive systems, which includes many ubiquitous and groupware systems. More specifically, I define

interactive systems as systems which end-users interact with directly, and which ask the user to make decisions about information disclosure.

2. Accessible to analysts with little experience in the method or privacy analysis

   If designing for privacy is to gain acceptance, the threshold for entry must be low as organizations will be reluctant to invest in specialized staff or training until a business case has been made. For these experiments the threshold will be set to what senior undergraduate computer science students can teach themselves based on available documentation. While strict, this is a reasonable approximation of what will be expected in a real-world setting.

3. Efficient in terms of the time-on-task needed to produce acceptable results

   To gain acceptance and entry, it is necessary to show that reasonable effort will result in reasonable results. Furthermore, it is necessary to show that these methods require little overhead (training and planning), and that they lead to the discovery and documentation of a significant set of problems.

1.2 Contribution Statement

There are three main contributions from this work. The first is to provide a deeper understanding of how users think about privacy, how they make decisions, and how current privacy management interfaces match or conflict with users' models. As part of this work, I have performed surveys and user experiments, and analyzed common interface design practices for privacy management tools. This information should provide valuable guidance to the development of future privacy-management interfaces.

The second contribution comes from the evaluation of current design and analysis frameworks, including their relative advantages and shortcomings. This evaluation includes an examination of the versatility and cost-effectiveness of some of these methods, the reasons for any relative advantage or disadvantage, as well as a discussion of how they integrate into existing software development practices. This evaluation will be important to our understanding of the state of the study of privacy in design, and the tools we have available. It will also enable us to argue about the appropriateness of these methods in the development of real-life systems.

Finally, STRAP itself is an important contribution both because of the analysis that went into its specification as well as the resulting framework and the relative advantages of its use when compared to other frameworks. As part of this process I demonstrate how structuring heuristic evaluation can lead to a more effective evaluation, especially when the system design or problem is preliminary or ill-defined. This should promote a re-evaluation of heuristic evaluation and its use in the study of different problem areas and at different stages of the design process.

1.3 Thesis Organization

The sections and chapters in this thesis are organized so as to best support and present the key claims and arguments rather than preserving the chronological order of the work done. Most importantly, part of the work presented in chapter four was carried out in parallel with some of the framework evaluations. The three studies presented in chapter four are also presented out of their chronological order.

This thesis document is organized as follows:

- Chapter two gives a general overview of the problem of privacy and its effect on the design of computer systems. This includes an overview of legislative and social developments as well as design methods and practices.
- Chapter three examines existing frameworks designed to specifically address privacy as a design requirement. This includes a description of each method, their theoretical foundations, as well as an analysis of the relative strengths and weaknesses of these foundations.
- Chapter four presents three studies exploring current privacy management systems and interfaces as well as user concerns and practices. The goal of this chapter is to compare and contrast the issues which existing frameworks emphasize with the kinds of problems users encounter.
- Chapter five builds on the two preceding chapters and describes a new analysis framework aimed at addressing some of the shortcomings identified in previous methods.
- Chapter six presents a number of design experiments designed to evaluate the different frameworks.
- Chapter seven is a discussion of the findings from the experiments presented in the previous chapter, the state of the art in privacy-aware design, legal requirements, and user needs.
- Chapter eight wraps with a discussion of questions raised and future work.

Finally, supporting materials are included for reference in the appendixes.

CHAPTER 2
BACKGROUND

Given the long history and public debate around the topic of privacy, including numerous literary works and legislative efforts dating as far back as 1361 (Langheinrich, 2001), and more than forty years of work on privacy in information systems, it is necessary to be selective in what is covered in this section. This section is not intended to be an authoritative history of privacy and the different types of work done in this area, but rather to give the reader the necessary background to understand the context within which this work is done, and the direction this work has taken.

This section is divided into four sub-sections. The first gives a brief definition of what privacy is, how our understanding of it has evolved, and why it is an important consideration in the design of information systems and their user interfaces. Next follows a brief discussion of the special importance that privacy concerns have on the development and deployment of interactive systems. The third section provides an overview of some of the different user-interfaces techniques used to address or alleviate privacy concerns in interactive systems. Finally, I present a high-level overview of the system development and design cycle, and the foundations on which the different analysis and design frameworks are based.

2.1 Introduction

Privacy is far from a novel concern. Thinkers as far back as Aristotle discussed the existence and need for a public sphere of political activity and a private sphere for

family and domestic life (Habermas, 1989). Privacy has continued to be a favorite topic for authors and philosophers through the ages, including George Orwell (1984, 1949), Ayn Rand (The Fountainhead, 1943) and Michel Foucault (The Subject and Power, 1983). These works are evidence of the rich and virtually universal concern with privacy throughout history and across cultures, and the public debate society has had about boundaries, what is acceptable and what is not. Other evidence of our complex and deep-rooted public debate about privacy is the long history of laws regulating, restricting, or protecting personal privacy, dating at least as far back as 1361 (Langheinrich, 2001)

Technology has been an important part of this dialogue since the end of the 19th century, with Warren and Brandeis' "The Right to Privacy" (1890). In this article, Warren and Brandeis, incensed by the privacy intrusions they saw made possible by the use of photographic technology; argued for the need for legal protection of the right to privacy. Incidentally, this article was published the same year Hollerith introduced the tabulator, a device aimed at preparing, sorting, and counting census returns, a pre-cursor to today's computers.

Computers did not become part of the privacy debate until the early 1960's with the introduction of the first commercially available computers, the IBM System/360 series, and its adoption by business and government agencies. While relatively late in the history of the privacy debate, the computer was still in its infancy when it became inextricably part of this debate. The dangers of this new technology, and its' potential impact on personal privacy had been most grimly illustrated through the Nazi use of the computers' precursor, the Hollerith tabulator in the 1930's to search census data, thereby efficiently identifying and rounding up persons of interest as part of their racial purification programs (Langheinrich, 2001, Black, 2001).

Computers changed the privacy debate because of their ability to collect, join, sort, store, and transmit records on a scale previously unimaginable. With the computer, one could for the first time imagine information not only persisting indefinitely, but also being copied endlessly and effortlessly, and searched and cross referenced efficiently and at will. Before the computer, data processing was as expensive or more than data collection, and privacy was ensured by economics. Data processing was only done selectively and with good reason due to the great expense involved. After the introduction of the computer, the cost of processing decreased dramatically, personal information became a commodity, valuable property which is sold and traded.

In response to these dangers, governments and international organizations started to debate the role of computer technology, the dangers it posed, and the ethical and legal guidelines for its use. In the US, the House of Representatives special subcommittee on Invasion of Privacy instituted hearings on this issue in 1965, coinciding with the Supreme Court ruling in Griswold v. Connecticut (381 U.S. 479, 1965), which formally introduced the concepts of “privacy” and the “right to privacy” as legal concepts in the US. Since then a long list of different laws, state, national and international, have been debated and enacted.

In the U.S. this debate took place against the backdrop of a tumultuous domestic situation, widespread paranoia about communist infiltrations, assassinations and an active domestic surveillance program. These events, combined with a long-standing distrust of centralized government, explain why people in the U.S. have traditionally been skeptical to government collection and use of personal information. The traditional target of privacy regulations in the U.S. has therefore typically been the government (US Privacy Act of 1974 (20 U.S.C. § 1232g, 1996). Corporate collection and use of personal

information has until recently been ignored in the U.S. and their practices unregulated. This laissez-faire approach to regulation is slowly changing with laws such as the US Health Insurance Portability and Accountability Act of 1996 (HIPAA) (US 1996), the US Gramm-Leach-Bliley Act of 1999 (GLBA) (US 1999).

In Europe, this debate has taken a different direction. Despite the continents' somewhat checkered history of authoritarianism and abuse of power, in Europe, most laws are aimed at regulating what information corporations can collect about individuals and how they may use that information. The government by default assumes the role of caretaker of citizens' privacy rights and is charged with enforcing laws and monitoring corporate practices.

Another important distinction between the European and U.S. views of privacy are the legal foundations or philosophies on which their laws are based. Until recently, privacy laws in the U.S. were based around the concept of private information as property, with laws formulated to restrict or regulate the exchange, sale and ownership of such information. In Europe, privacy laws are built on the concept of privacy as a human right, and laws are formulated with the intent of providing blanket protection. As L. Jean Camp (1999) explains, this has led to very different approaches and legislative approaches:

> *"The American tradition of concern for privacy varies from the European approach. The European Community and Canada have principles of data protection, whereas the American tradition revolves around privacy. American considerations are based on common law tradition and a constitutional right, rather than on the more practical approach implied by data protection.*

Privacy law, as opposed to data protection, has been implemented piecemeal. Privacy protection views each subject area of data as separate and requires action for each subject area as necessary. The data protection approach offers blanket guidelines for all data with an identifiable subject."

These philosophical differences between Europe and the U.S., while influenced by the different historical and political climates in which the debate has taken place, are also caused by lack of a single clear definition of what "privacy" actually is. This lack of definition is especially striking given the long history of discussion, debate, and regulation of privacy. There are two reasons for this. The term privacy has been used in many different contexts, often as an emotional modifier to lend urgency and weight to problems and arguments. Second, as Palen and Dourish (2003) note, there is tremendous variance in what people consider to be sensitive and private, and what they consider an invasion. Not only is there significant variability between subjects, but also within subjects; privacy is locally negotiated, and decisions are often made on the spot and are difficult to predict and inconsistent. It is therefore not surprising that the term "privacy" itself, and what constitutes an invasion of privacy, is inherently difficult to define. This having been said, within the context of computing, there are three common definitions in use:

1. The right to be left alone without unwarranted intrusion by government, media or other institution or individuals (Brandeis & Warren, 1890)
2. The right to information self-determination (Westin, 1967)
3. The right to protect personal data and communication (Goldberg, 1997)

The first of these definitions of privacy is the most basic, the ability to define a private sphere and have that sphere protected from intrusion from whomever we choose. This definition was the basis for most early legislative efforts (such as US Privacy Act of 1974), and is still influential in shaping public opinion and legislative efforts over issues such as spam and who has the right to contact us.

The second definition is a super-set of the first and introduces the notion of ownership, and the right to control information even after disclosure. It argues that information is a commodity, but that some ownership always resides with the data subject and does not fully transfer with disclosure or discovery, and that subsequent access and use of that information is subject to the data subjects' approval. This is a more restrictive definition for data processors, but one that is gaining increasing acceptance among regulators and forms the foundation for such laws as HIPAA (US 1996), GLBA (US 1999), and the EU privacy directive (EU, 1995).

The last of these definitions is one adopted by many security researchers and advocates. They argue that every individual not only has a right to privacy, but also a right to protect and safeguard themselves and their information. Only through the use of secure communication, systems, and transactions can we really protect our privacy. This definition gained popularity in the nineties because of efforts by the U.S. government to limit and regulate the use and export of end-user encryption programs such as PGP (Garfinkel 1995) and their promotion of the Clipper chip (Denning, 1993) and DSS (NIST 1994). Opposition to these proposals concentrated on these claimed rights of the individual to safeguard their information.

These three definitions are very different, but complementary, and all three operate simultaneously in the public mind. When combined, these three definitions form

the foundation for the work presented in this thesis: *Individuals have the right to determine what is public and what is private, and to whom. They own the rights to their own information, including the right to determine how that information is collected and used. Finally, they have the right to take steps to protect their own privacy and ensure that third parties live up to their agreements.* Some of the key challenges to realizing this vision of privacy management are fundamental HCI challenges: how to support information and situation awareness so that users can make informed decisions, how to provide control mechanisms which allow users to efficiently and effectively control their information, and how to provide these capabilities without overwhelming end-users.

2.2 Privacy in Interactive Systems

Privacy is a serious concern for many different types of applications and uses of computers. It is therefore not surprising that privacy has been studied in a number of different contexts in computing, from computer-supported collaborative work (CSCW) to Ubiquitous computing, from mobile devices and services to online communities, and from e-commerce to data-mining and healthcare informatics. This list continues to grow as the number of applications take advantage of or rely on collecting, sharing or processing potentially private information with the proliferation of sensors, network connections and available processing power.

In this work I restrict myself primarily to the problems posed in interactive systems, which I define as systems with which end-users interact directly, and which ask the user to make decisions about information disclosure and use. This is a pretty flexible definition, incorporating many different types of systems. I make this restriction not because the majority of violations occur in the context of such systems, or that these

systems pose the greatest dangers to users, but because of my interest in to the HCI challenges associated with the design of these systems. Another reason for choosing this set of systems is that privacy problems and concerns have been shown to be major obstacles to the introduction and adoption of such systems, from e-commerce (Jupiter 2002) to groupware (Grudin 1994), and mobile location services (Smith et al. 2004) to name just a few.

Privacy issues and their impact on system adoption and design have perhaps been most extensively studied in the context of groupware or CSCW systems, which often by their nature require some sharing of personal information by the individual for the benefit of a larger group, be it in order to promote awareness of coworkers' activities, providing context for collaborative activities, or promoting distributed collaboration. This sharing of personal information comes at a price. Grudin (2001) notes that while groupware introduces new possibilities, it de-situates users, leading to a potential breakdown of established social norms. Identity, location, and activity are examples of common information types that can both facilitate group work and expose users to potential privacy invasions.

In ubiquitous computer systems, the prevalence of sensing and recording systems as well as extensive use of potentially private and sensitive information, possibly without the users' awareness or consent, raises critical privacy concerns. Bellotti and Sellen (1993) stress the importance of feedback and control over information capture, access, purpose, and construction. Similarly, Abowd and Mynatt (2000) describe the challenges of designing collaborative environments where actions and roles are dynamic.

Privacy has also been an important concern in groupware calendar systems, most extensively studied in the work of Grudin and Palen (Grudin 2004; Grudin and Palen,

1997; Palen, 1999; Palen and Grudin 2002). While their studies were not specifically aimed at studying privacy, they found that privacy concerns significantly affected adoption, use, and evolution of groupware calendar systems.

The success of groupware systems depends in part on the degree to which the individual benefits of contributing outweigh its costs. Palen and Dourish (2003) describe these tradeoffs as the resolution of tensions between the need to be part of the world and receive some benefit, and the need to shield and protect oneself and ones personal life. In the case of calendar systems, Grudin (1994) writes:

> *"Just as people who live in buildings with paper-thin walls may adopt a convention of ignoring what they cannot help overhearing, people who allow open access to their calendar details assume that people will access information only when needed and would be offended by an inquiry that revealed "snooping." Being able to block off a calendar entry or reserve a conference room is deemed an adequate balance. Privacy is ultimately a psychological construct, with malleable ties to specific objective conditions."*

While workplace habits and individual experience may motivate users to share their schedules, mechanisms to manage and protect ones privacy must be present. The frequency with which such mechanisms are used seems to be less important than the fact that they exist, and the impact they have on risk perception. Such mechanisms may be as simple as the ability to omit sensitive events, give cryptic or context-sensitive names to events, to enable reciprocity of access settings, or to schedule defensively to regulate interruption (Palen, 1999).

2.3 Privacy as a Design Problem

A number of different approaches have been adopted in an attempt to address privacy concerns in different types of interactive systems. These range from the simplest, posting notices and reassurances, to designing interfaces and systems to specifically encourage specific expectations and behaviors in users. Encouraging the adoption of existing and acceptable social norms has long been an important considerations in the design of online environments (Bruckman & Resnick, 1995), ubiquitous computing systems (Dourish, 1993), and computing systems in general (Lessig, 1999).

Carrying over or encouraging the adoption of existing social norms has been challenging in many cases. Grudin (2001) argues that in many computer systems, and especially groupware systems, users are de-situated and disembodied. By this, Grudin means that most systems make other users faceless and anonymous, and give the user a sense of anonymity which desensitizes them to the social or economic costs or consequences of their actions. This desensitization in turn increases the likelihood of a breakdown in social norms and what is considered responsible and appropriate behavior. In some environments, such as many online communities, there are relatively simple techniques for doing so. Both Bruckman in MediaMOO (1995) and Horn in ECHO (1998) tied users' online identities to their real-world identities to encourage or instill a sense of accountability and set reasonable expectations as to what would and would not be acceptable behavior. The subsequent adoption of appropriate norms not only decreases the rate of inappropriate behavior, it also allows users to more efficiently police the system and each other, eliminating or reducing the need for a central authority.

These ideas were explored further by Erickson and Kellogg (2000) in their work on social translucency and trust-building by supporting social accountability. In their

system, Babble, they showed that when users' actions were made visible this promoted trust and accountability. In Babble, mechanisms were implemented to allow users to see how others had acted in the past. By tracking and making others' actions visible, and thereby providing a mechanism for holding others accountable, and letting users know that such mechanism exist, greatly discouraged inappropriate behavior, such as eavesdropping and monitoring. In Babble, people reverted and more closely adhered to established offline social norms, and occurrences of inappropriate behavior declined.

Providing feedback and making actions visible are of course established usability principles (Norman, 1988). Traditionally these are used to promote understanding and boost user confidence in the system by showing that it is doing what the user intended. For privacy, these techniques are most commonly used to ensure users know what information or image they are projecting to others, hopefully preventing unintended disclosures or embarrassing misunderstandings. Most typically we see these mechanisms in instant messaging and chat environments, such as AT&T's Hubbub (Isaacs et al. 2002) and Microsoft's V-Chat and Comic Chat systems (Cheng et al. 2002) where users see their own avatar as others do.

Naturally, any successful systems must build upon an understanding of users' expectations, beliefs, knowledge, and mental models with regards to privacy. We know from a number of surveys that people consider privacy to be important (Culnan 1999, Earp & Meyer 2000, Kobsa 2002). Privacy concerns are the most cited reasons for avoiding the use of e-commerce systems, an aversion that industry groups estimate costs e-commerce companies USD 25 billion per year in lost revenue opportunities per year (Jupiter 2002). Most surveys have found that people are more concerned about their privacy online than offline (Jensen et al. 2005), even though most cases of identity theft

occur offline (Javelin 2005). It is not surprising that industry groups invest significant resources to build consumer confidence and engage in voluntary efforts such as publishing privacy policies and seeking certification from organizations such as TRUSTe.

Turning to management interfaces and mechanisms we find that the most commonly used information dissemination method is the almost ubiquitous privacy policy or statement. These policies are often the only source of information a user has about a company's privacy practices, and they detail the options users have with regard to participation and controlling information use. We know from surveys that though users think it is important for sites to present such policies, they are less than impressed with their quality and accuracy (Culnan & Milne 2001). Surveys show that users find privacy policies to be boring, hard to read and understand, hard to find, and that they don't answer the kinds of questions they are interested in (Culnan & Milne 2001). This mismatch between user concerns and the focus of policies has more extensively been studied by Earp et al. (2005).

While most surveys report that a sizable portion of users claim to read such policies or notices regularly (Culnan & Milne 2001, Kobsa 2002), there is evidence to suggest these reports are greatly exaggerated (Jensen et al., 2005). The reasons for this discrepancy between observed and reported behavior are not entirely clear, though it is likely that subjects are to a certain extent trying to rationalize their own behavior. Users know that the responsible thing to do would be to read these policies and therefore tend to over-report these practices. The reasons for why so few people consult policies is clearer, including the fact that policies do not focus on the issues users care about and the use of intimidating or difficult language in these policies (Jensen & Potts 2004).

To overcome some of the problems associated with privacy policies and reduce the burden on users, machine-readable policy specification languages, such as P3P (Cranor et al., 2002) and EPAL (Ashley & Schunter, 2002), have been proposed. These policies can be read by automated agents (such as privacy critics (Ackerman & Cranor, 1999) and Privacybird (Cranor et al. 2002)), only alerting users if the policy is likely to cause concern. The theory is that by filtering out the noise and drawing users' attention to only those policy elements which require attention, users are more likely to be engaged.

Privacy policies are important not only because of their widespread use, but because of their impact on system design. The relationship between policy and code was explored by Lessig (1999), who described source-code as an instantiation of policy. Programs are a set of rules which the computer, and by extension the user, must follow in order to accomplish a set of goals. Some of these rules are intentional, some result from ad-hoc and potentially inconsistent implementation decisions, while others are side-effects of tools and infrastructure used during system design and at run-time. Privacy policies are laws and rules which must ultimately be implemented as working code. These rules are typically written by lawyers or managers, and system designers and coders are responsible for interpreting and implementing as best they can.

The danger of this procedure is that promises or guarantees may be made which cannot be enforced in code, or that as the system and policy evolve discrepancies can emerge. For these reasons efforts have been made to make policy more directly connected with implementation. IBM, among others, have been working on using EPAL to evaluate and modify database queries in such a way as to ensure an application does not violate policy (Bohrer 2003). By decoupling the code from the policy, either can be

modified at any time without affecting the other, making it trivial to ensuring policy compliance in these applications.

2.4 Design of Information Systems

The main focus of this thesis is to examine current design and analysis methods aimed at addressing privacy, how they compare in terms of the types of problems they identify, how much effort and experience they require from an analyst, and how they may be improved. As such it is crucial to discuss what design is in the context of this work, the processes of design and how privacy fits as a design problem and into the overall design process and current design practices.

The term design has been used in many different and not always complimentary contexts. In this work I am not concerned about design in the aesthetic sense (graphical design), or even the more low-level user interface design processes (selection and placement of interface components). In the context of this work, design refers to the process of determining what a system should do, how the process should be accomplished, and the information the system needs to give the user in order for the process to work smoothly. This definition of design has been a focus of interest for both software engineers and usability engineers, and is sometimes referred to as interaction design (Preece et al. 2002).

In the following sections I will focus on giving an overview of the different models for the design process in order to discuss where privacy considerations need to be taken into consideration. I then discuss the different methods proposed for dealing specifically with privacy.

2.4.1 Design Process

Over the years a number of major lifecycle-models have been proposed and adopted, most of these originating from the software engineering community. The earliest widely-adopted model was the waterfall lifecycle model (Royce, 1970). This model defines the design process as consisting of a linear 5 stage model, with iteration contained to adjacent levels in the design process (see figure 1).

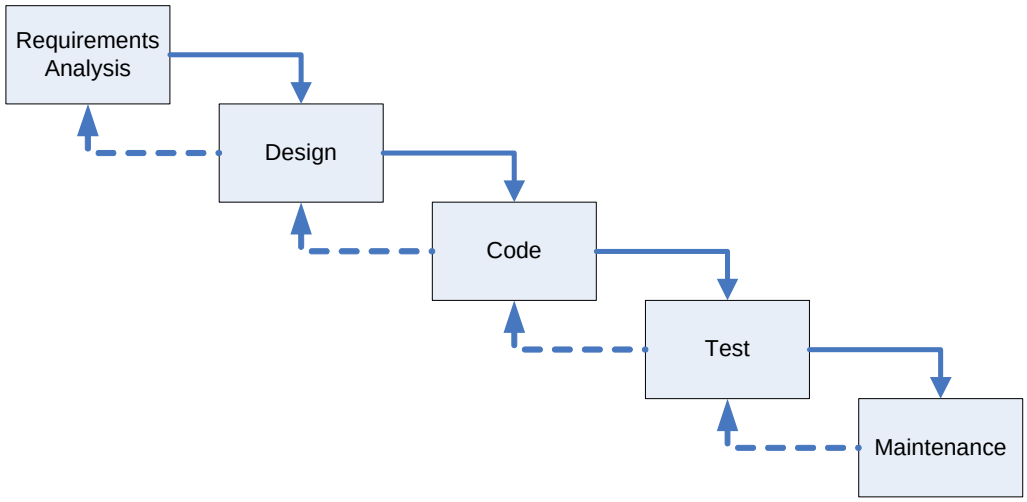


Figure 1: Waterfall Life-cycle Model (Royce 1970)

The major drawback of this model is its rigidity and inability to accommodate iteration. Software development was assumed to be done in defined stages, with fixed deliverables at the end of each stage. This model started to break down with the emergence of smaller, more interactive teams, and the need to involve the user in the design of systems. Since then a number of different models have been proposed, including the spiral lifecycle (Boehm, 1985), the Star lifecycle (Hartson & Hix, 1989), and the Usability Engineering lifecycle model (Mayhew, 1999).

Though different, these models follow a general high-level pattern: Systems requirements are derived, typically from careful studies of user activities and needs. This

data, often in the form of thick ethnographic descriptions or studies must be analyzed and structured in order to identify salient practices. From these an analyst derives requirements for a solution, and these are in turn used to derive design alternatives and implement competing prototypes. Finally, some evaluation is done of the resulting systems.

For each step in these design processes there are different techniques aimed at helping designers and analysts accomplish their task. Again, these come both from the HCI and software engineering literature. A comprehensive survey of these methods is beyond the scope of this work.

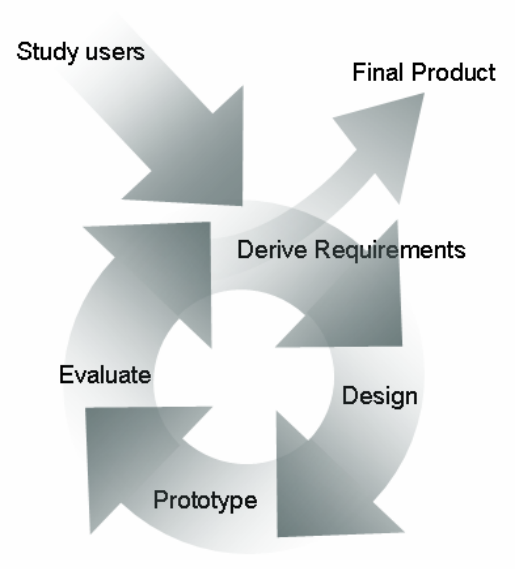


Figure 2: Simple Iterative Design Model

While each of the above-mentioned models has its uses and followers, many researchers and designers are adopting simpler iterative models more suitable for rapid prototyping. These models typically contain all the steps descried above, but in a cyclic process where evaluation, rather than being the final step, is used to refine the understanding of users and their needs and restarting the entire process (figure 2). This model is similar to that discussed by Preece et al (2002). Because of its simplicity and

general appeal, I use this model later when discussing the different design methods and what parts of the design process they support.

2.4.2 Foundations of Privacy-Aware Design

Whether it is called ‘privacy-aware design’, ‘privacy by design’, or ‘designing for privacy’, a number of different design and analysis frameworks have been proposed to help designers both identify and address privacy problems in applications. These are very specific, specialized analysis methods, and are not meant to eliminate the need for other development or analysis methods. For instance, though many privacy-aware design methods examined in this paper come from the human-computer interaction community, they do not focus on traditional usability concerns. Therefore, their use does not eliminate the need for a thorough analysis of usability concerns, but rather augments them by helping to identify a new set of concerns.

The vast majority of these frameworks have taken the form of guidelines and heuristic-based frameworks, and come from either the usability or software engineering communities. In this section I will give an overview of the guidelines proposed by government and international organizations followed by an examination of the basis for the use of heuristics in usability and design before briefly discussing the different design frameworks which have been proposed.

2.4.2.1 Government Guidelines

The earliest design guidelines to address privacy concerns were developed by government sponsored panels. This is not surprising given that these started taking place very early in the history of computing when such technology was almost exclusively in the hands of government agencies and major corporations

The earliest set of privacy guidelines, the Fair Information Principles (FIPs) (HEW, 1973) have evolved over the years, and are still in use. These guidelines were derived from expert testimony by the U.S. Department of Health Education and Welfare's (HEW) Advisory Committee on Automated Personal Data Systems (also known as the HEW report). This report set ethical guidelines for developers, designers, and corporations, the latest revision of which was released by the Federal Trade Commission (FTC) in their 2000 report on online privacy (FTC, 2000) (Figure 3). This last revision simplified the FIPs from five to four design principles.

1. **Notice/awareness** – *Consumers should be given notice of an entity's information practices before any personal information is collected from them.*
2. **Choice/Consent** – *Giving consumers options as to how any personal information collected from them may be used.*
3. **Integrity/Security** – *Data should be accurate and secure.*
4. **Enforcement/Redress** – *Privacy protection can only be effective if there is a mechanism in place to enforce them.*

Figure 3: Fair Information Practices (FTC, 2000)

Though groundbreaking, these guidelines are fairly high-level and abstract. This lack of specificity and detail may make these guidelines difficult to apply in the real world and may explain why only a handful of websites among those surveyed by the FTC were in compliance with these principles (FTC, 2000).

Though the FIPs have likely been the most influential guidelines, at least in the U.S., they are not the only ones to have been proposed. The Organization of Economic Cooperation and Development (OECD) made its own set of recommendations in 1980. This is a longer list of guidelines (see figure 4), and markedly more aggressive in their stance for privacy rights. The OECD guidelines are not inconsistent with the FIPs, and can in be seen as a superset of these.

1. **Collection Limitation** – *There should be limits to the collection of personal data and any such data should be obtained by lawful and fair means and, where appropriate, with the knowledge or consent of the data subject.*
2. **Data Quality** – *Personal data should be relevant to the purposes for which they are to be used, and, to the extent necessary for those purposes, should be accurate, complete and kept up-to-date*.
3. **Purpose Specification** – *The purposes for which personal data are collected should be specified not later than at the time of data collection and the subsequent use limited to the fulfillment of those purposes or such others as are not incompatible with those purposes and as are specified on each occasion of change of policy.*
4. **Use Limitation** – *Personal data should not be disclosed, made available or otherwise used for purposes other than those specified in accordance with [the stated purpose] except with the consent of the data subject; or by the authority of law.*
5. **Security Safeguards** – *Personal data should be protected by reasonable security safeguards against such risks as loss or unauthorized access, destruction, use, modification or disclosure of data.*
6. **Openness** - *There should be a general policy of openness about developments, practices and policies with respect to personal data. Means should be readily available of establishing the existence and nature of personal data, and the main purposes of their use, as well as the identity and residence of the data controller.*
7. **Individual Participation** – *An individual should have the right*
   a) *to obtain from a data controller, or otherwise, confirmation of whether or not the data controller has data relating to him;*
   b) *to have communicated to him, data relating to him within a reasonable time; at a charge, if any, that is not excessive; in a reasonable manner; and in a form that is readily intelligible to him;*
   c) *c) to be given reasons if a request made under subparagraphs(a) and (b) is denied, and to be able to challenge such denial; and*
   d) *d) to challenge data relating to him and, if the challenge is successful, to have the data erased, rectified, completed or amended.*
8. **Accountability –** *A data controller should be accountable for complying with measures which give effect to the principles stated above.*

Figure 4: Protection of Privacy & Transborder Flows of Personal Data (OECD, 1980)

It is important to note that the OECD guidelines were not specifically created for designers, but rather as a framework for lawmakers in an effort to streamline cross-national business and data transfers. The OECD guidelines were not only influenced by the FIPs but also by existing national privacy legislation in Europe, which any corporation or government transmitting personal information across their borders would have to deal with.

Since then there have been a number of laws and frameworks proposed and passed, including the EU Privacy Directive (EU, 1995), the US Health Insurance Portability and Accountability Act of 1996 (HIPAA) (US 1996), the US Gramm-Leach-Bliley Act of 1999 (GLBA) (US 1999), the US Children's Online Privacy Protection Act of 1998 (COPPA) (US 1998), and the US Safe Harbor framework of 2000 (Long & Pang Quek, 2002). Corporations, organizations, and ultimately designers and developers have increasingly had to deal with this diverse and at times inconsistent set of laws and legal frameworks in the design of information systems. These frameworks impose requirements on the kinds of systems which may be built and used, with potentially severe penalties for infractions.

While it is necessary for designers and developers to know what laws they are subject to, and what requirements these impose on any information system or process, existing frameworks and laws provide very little support for informing the design or evaluation process. Furthermore, it is not always possible to know a priori which of these frameworks or regulations an application may be affected by. Decisions about where to market an application and to what segments are often made independently of the development process, and the legislative process is continuously evolving.

Even when the legislative context is well defined, legal documents often use vague terminology, especially in the US, where the exact definitions are often left to the courts. Given the complexity of the legal landscape, and the often severe penalties associated with violations, either imposed by the courts or in the form of negative publicity, it can be assumed that many if not most violations are accidental rather than intentional. This argues for the need for a set of tools for designers to more carefully consider and take these types of requirements into account in the design process.

2.4.2.2 Heuristics in Usability

The appeal of guidelines and heuristic in design and evaluation is that they are relatively simple to understand and learn, and relatively quick and inexpensive to apply. This is important: To influence everyday design practices, it is not enough to show that a method or technique is effective but that the ramp-up costs are surmountable, and that the return on investment is significant. Heuristics and guidelines typically score highly in this respect. Because both guidelines and heuristics are often derived from real-world experiences, they also have great external validity and appeal.

Nielsen & Molich (1990), with their study of the heuristic evaluation of user interfaces, were the first to bring this form of evaluation into the mainstream of usability engineering. Heuristics and guidelines have of course been age-old tools in design, but Nielsen and Molich showed how a group of analysts, using artifacts to guide them (screenshots or prototypes) could perform as effective an analysis as designers using more formal methods like GOMS, at a fraction of the cost.

Nielsen's list of heuristics was derived from the examination of hundreds of sometimes conflicting design guidelines. Keeping the list of heuristics down to a

manageable and easily memorized number of guidelines was deemed essential. Nielsen argued that a designer would be more successful applying a small set of carefully selected guidelines than a much long but more complete list, and that the end result would likely be better (Nielsen, 1994). Nielsen has made numerous revisions to the list of heuristics for general usability evaluation; an overview of the current list is available in figure 5.

These heuristics are intended to guide designers in evaluating and analyzing prototypes and design solutions. To a lesser extent, they can also guide the creation of solutions. Given that this list is what solutions will be evaluated against, designers can use them to inspire solutions (by dictating additional requirements), or by cutting short design paths which will obviously lead to poor solutions (according to the list of heuristics).

Nielsen and Molich were primarily concerned with general system usability, evaluating how intuitive and user friendly a system would be to use. They were not concerned with privacy or designing for privacy awareness. Though their heuristics do have some bearing on privacy awareness, relying solely on their heuristics for this purpose would likely prove too difficult and inefficient. A more effective and desirable solution would be to define a set of guidelines and heuristics to specifically address and identify the kinds of problems seen in designing for privacy.

1. **Visibility of system status** – *The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.*
2. **Match between system and the real world** – *The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.*
3. **User control and freedom** – *Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.*
4. **Consistency and standards** – *Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.*
5. **Error prevention** – *Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.*
6. **Recognition rather than recall** – *Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.*
7. **Flexibility and efficiency of use** – *Accelerators - unseen by the novice user - may speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.*
8. **Aesthetic and minimalist design** – *Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.*
9. **Help users recognize, diagnose, and recover from errors** – *Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.*
10. **Help and documentation** – *Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.*

Figure 5: Usability Heuristics (Nielsen, 1994)

Over the years, a number of design and analysis frameworks for privacy have been proposed. The first “real” framework in the sense that it provides more than a set of guidelines, and the most influential, has been the framework proposed by Bellotti and Sellen (1993). Apart from being the first example of a design framework developed specifically to address privacy in the design process, this framework is significant because it also started a trend of using heuristic-based frameworks to address privacy as a design consideration. While different from Nielsen and Molich’s work in focus and the specific heuristics used, these methods build on the foundations which Nielsen built by demonstrating the efficiency of heuristic evaluation.

The Bellotti & Sellen framework was designed specifically to deal with the privacy concerns emerging from the design of “Media Spaces” and ubiquitous environments (Bly et al. 1993). This has continued to be a trend, with most subsequent frameworks emerging from the same research area, and typically being based around the use of heuristics. These frameworks include Hong et al. (2004), and Langheinrich (2001). There have also been frameworks coming from the software engineering literature which are not based on heuristics, most notably the i* framework (Yu & Cysneiros, 2002), and to a certain extent the Patrick and Kenny (2003) framework. These frameworks are discussed in detail in the next chapter.

CHAPTER 3
DESIGN FRAMEWORKS FOR PRIVACY

Five major design and analysis frameworks have been proposed specifically to deal with privacy issues, those of Bellotti & Sellen (1993, later Bellotti 1997), Langheinrich (2001), Hong et al. (2004), Patrick & Kenny (2003), and the i* framework by Yu & Cysneiros (2002). These frameworks are considered to be the best known, the most influential on other research and development, the most complete, or the most innovative. In this section I will first describe each of these frameworks and the foundations and assumptions they build on, and then analyze their relative strengths and weaknesses, as well as how they fit into the overall system design process.

3.1 The Five Major Design Frameworks

The first framework aimed specifically at dealing with privacy in the design of interactive systems was that proposed by Bellotti and Sellen (1993). This framework was specifically developed to deal with the privacy concerns emerging from the design of ubiquitous computing environments such as the Rank Xerox EuroParc's Media Spaces (Bly et al., 1993). As such, it was also heavily influenced by the types of systems, ubiquitous systems and media spaces, which the authors studied in the process.

In this framework, Bellotti & Sellen presented a set of steps for designers to follow in order to evaluate a system or design, and a set of heuristics to suggest and evaluate solutions. Because this work was continued by Bellotti, this section will discuss

and analyze the framework as it evolved and was presented in the most recent publication (Bellotti, 1997), and a brief overview is provided in Figure 6.

The designer asks a set of questions about the proposed system meant to identify potential privacy problems:

- What information is captured and how?
- What happens with this information?
- How is this information made accessible to the user?
- What is the purpose of the information collected?

Given the answers to these questions, the designer identifies problems and ways to address them. The following criteria are both guidelines for desirable design, and benchmarks for evaluating potential solutions:

- Trustworthiness
- Perceptibility
- Minimal intrusiveness
- Flexibility
- Meaningfulness
- Low cost
- Data quality
- Use limitation
- Openness
- Accountability
- Appropriate timing
- Unobtrusiveness
- Fail safety
- Low effort
- Learnability
- Collection and limitation
- Purpose specification
- Security safeguards
- Individual participation

Figure 6: Bellotti Framework (1997)

The Bellotti & Sellen framework combines heuristics with a simple analytic procedure. These heuristics were chosen from observations of the kinds of issues that emerged in their development and deployment of a multi-site, always on, shared video and audio link system (Bly et al., 1993). In these types of systems, privacy problems are most likely to emerge from the automated capture, storage and processing of information about the users' everyday activities. Designers are therefore encouraged to ask key questions regarding how and what and for what purpose the system captures information. These questions are designed to alert designers to potential privacy problems caused by the way the system is designed. These problems are then addressed and evaluated though the use of heuristics, of which Bellotti & Sellen identified 19.

This work came very early in the study of ubiquitous computing and groupware systems and was warmly received by the HCI and research communities. Yet, despite

having been widely cited and read, this framework seems to have had relatively little impact on current design practice. There are no examples in the literature of other groups or researchers using this framework in the development, analysis or evaluation of systems. While not an uncommon fate for academic research, it is somewhat puzzling given the amount of attention and effort that has been invested on the topic of privacy.

Part of the reason for this lack of adoption may be that this method was never evaluated in terms of efficiency, effectiveness, or applicability. This is a common problem for all five of the frameworks reviewed in this section, and indeed much of the work in the design literature, both on in HCI and software engineering. The absence of such data makes it difficult, both for developers and researchers to justify spending the time and effort to perform this type of analysis. Another important thing to note is that Bellotti & Sellen never made any claims about the applicability of their method outside the design of ubiquitous systems such as Portholes (Bly et al., 1993). Such a limited scope coupled with a more thorough evaluation of the method may have limited its' appeal in the eyes of some.

While the Bellotti & Sellen framework was the only point of reference in terms of frameworks for almost a decade, there has recently been increased interest in the development of new frameworks for privacy-aware design. Interestingly enough, most of these efforts have originated within the same ubiquitous computing community (including Langheinrich, 2001, and Hong et al, 2004). Because these frameworks have been around a relatively short time compared to the Bellotti & Sellen framework, it is difficult to judge whether they will have more of an impact on the design and research communities.

The framework which most closely resembles that of Bellotti & Sellen is the 'Risk Model' framework developed by Hong et al. (2004) (see figure 7 for an overview). Like the Bellotti & Sellen framework, the Risk Model framework is based around the use of a list of analytical questions and heuristics, and has been derived from an examination of the different systems developed and deployed within a specific research group.

More specifically, the Risk Model framework contains three steps. The first step is to perform a risk analysis in which the designer asks a set of analytic questions of the proposed system. In this case the questions are divided into two categories; social and organizational questions, and technology related questions. This method goes to eleven multi-stage questions from Bellotti & Sellen's original four. As the two categories of questions imply, the authors want the analyst to consider the social context within which the system will be used, as well as the more technical details of how, when, why and for whom the system will be capturing information.

Curiously enough, this framework does not ask the designer consider a very basic question, namely what information is captured. This is a very important question to consider, one which will greatly affect what concerns users have with regard to the system, and a natural part of any design analysis. Bellotti & Sellen did include this in their four analysis questions, and one can only assume that its omission from the Risk Model framework is an oversight.

The designer starts by asking a set of analytical questions, a social and a technical set:

Social and Organizational Context

- Who are the users of the system?
- What kinds of personal information are shared?
- What is the value proposition for sharing personal information?
- What are the relationships between data sharers and data observers?
- Is there the potential for malicious data observers
- Are there other stakeholders or third parties that might be affected?

Technology

- How is personal information collected?
- How is personal information shared?
- How much information is shared?
- What is the quality of the information shared?
- How long is personal data retained?

A problem should only be remedied if the cost is lower than the product of the likelihood of a violation and the damage it would cause, providing a set of priorities. A set of questions guide the discovery of potential solutions:

- How does the unwanted disclosure take place?
- How much choice, control, and awareness do data sharers have over their personal information? What control and feedback mechanisms are there?
- What are the default settings? Are these defaults useful?
- In what cases is it easier, more important, or more cost-effective to prevent unwanted disclosures and abuses? Detect disclosures and abuses?
- Are there ways for data sharers to maintain plausible deniability?
- What mechanisms for recourse or recovery are there if there is an unwanted disclosure or an abuse of personal information?

Figure 7: Hong et al's Risk Model framework (2004)

Once potential problems have been identified and catalogued using the risk analysis method, the designer must examine them using the risk management method. In this phase, potential problems are evaluated based on their potential for harm (financial costs and liabilities to the data collector) and the likelihood of occurrence. These potential damages are used to rank the severity of a problem, and are then compared to the estimated cost of fixing or avoiding the problem, which include design and programming hours, or lost revenue from reduced system functionality. If the potential damages of any one potential problem outweigh the cost of addressing it, the fix should be undertaken. These numbers need not be accurate: *"[…] the utility of this cost-benefit analysis comes not so much from accurate and precise values, but from having the design team think through the issues of likelihood, damage, and cost […]"* (Hong et al., 2004)

Once the potential problems are weighted and prioritized, a series of questions are used to identify and evaluate potential design solutions rather than a set of heuristics. These sets of questions, like Bellotti & Sellen's heuristics, were derived from the study and development of ubiquitous systems. What is interesting about these questions is that in many cases it would be necessary to ask a number of these before one could make decisions about the potential damages or risks associated with a vulnerability. For instance, how an unwanted disclosure takes place is an essential piece of information in this evaluation, as considerations about what potential problems to avoid or address.

Turning our attention to the part of the method referring to the weighing and prioritizing of privacy and security flaws, it is worth noting that this was first proposed by Butler and Fischbeck (2002), and makes sense from a return-on-investment perspective. As Hong et al. argue, the value of this step is in directing attention to the need or value of prioritizing problems rather than providing an algorithmic and objective

method for determining priorities. The weakness of these models is that they rely on estimates of the likelihood of a fault, for which there is little support or methodology. These methods also emphasize the costs to the data collector rather than that of the data subjects, and are largely incapable of dealing with conditional probabilities and cumulative costs.

The likelihood of an incident occurring as well as estimated costs are typically calculated based on previous incidents. For certain types of applications, those for which a company or designer has built a knowledge-base, these estimates may be fairly accurate. It is however in the nature of these types of problems to be unpredictable, and for many problems designers will be left "guestimating." The cost of addressing a problem is easier to estimate, using standard estimates of the number of programmer man-hours, legal penalties, or lost productivity due to limited functionality.

A more serious shortcoming of these methods is their inability to account for conditional probabilities and cumulative damages frequently associated with privacy and security vulnerabilities. By conditional probabilities I refer to the problem that critical and highly sensitive systems are often breached or attacked from less important and vulnerable sub-systems. In other words, suffering a relatively low-cost but high-likelihood attack can and often does increase the likelihood of more critical systems being breached. Cumulative costs are conceptually similar. The loss or compromise of one type or set of data, such as social security numbers, can greatly increase the costs or privacy risks associated with the loss of other, relatively benign types of data, such as the subjects' name, date of birth, mothers' maiden names or place of birth, typically used as secondary identifiers in economic transactions.

While these are important limitations of these approaches, they can be useful in helping designers and analysts prioritize and focus their efforts. The real danger with these estimates is that they encourage the use of concrete numbers, which may make these estimates and rough classifications seem firmer than they really are. To avoid this type of 'fixation,' many bug-tracking systems instead use a simpler classification scheme where analysts directly assign each bug a simple priority classification (i.e. 1-10, or low, medium, high) and decide that all problems of a certain priority must be addressed, and that all issues under a certain priority will only be addressed is the cost is under a given threshold. Though not substantially different from the method advocated here, using a less quantitative approach may help emphasize the subjective nature of the data.

The two preceding frameworks were based on observations of the design and deployment of working ubiquitous systems, albeit a limited number of them. Real-world experience is of course an invaluable source for knowledge and lessons about best practices, but it is not the only source for heuristics. Another important source has been the legal frameworks presented in section 2.4.2.1. These guidelines have been the starting point for two frameworks, those of Langheinrich (2001) and Patrick & Kenny (2003).

Langheinrich's framework (Langheinrich, 2001) differs from the other frameworks in that it does not provide any support for the analysis of where problems may emerge. In his paper, Langheinrich instead presents a through and thoughtful analysis of the philosophical, historic, and legal context and considerations for designers and application developers, especially for those of ubiquitous computing systems. Langheinrich presents six design principles, derived from the Fair Information Practices (FIPs) (Figure 3, page 26). These principles are: notice, choice and consent, proximity and locality, anonymity and pseudonymity, security, and access and recourse.

The only major addition to the FIPs which Langheinrich proposes is that of considering proximity and locality, and anonymity and pseudonymity. These two principles are both aimed at dealing with the kinds of privacy problems most often encountered in ubiquitous systems, which is to limit the scope of the application or system, either geographically, or in terms of its' accuracy and knowledge. Langheinrich argues that many of the privacy problems encountered in ubiquitous applications can be resolved if anonymity or at least pseudonymity can be ensured. Furthermore, by limiting data capture and access geographically or by context, he argues that many privacy concerns can be avoided.

The value of this framework is the depth of analysis on which it is based, and the way it helps ground the more general-purpose FIPs to the domain of ubiquitous computing. The major drawback to this framework is its lack of a method or procedure to identify potential problems in an application or design. While such a framework may help designers get into the right mindset, or deal with the more obvious privacy problems, it does not help them identify and address the more complex and hidden problems and faults, thereby limiting its' appeal and usefulness.

The second framework based on legal frameworks and requirements is that by Patrick & Kenny (2003). They base their framework on the OECD guidelines (figure 4, page 27), and an analysis of the usability requirements these imply. These usability requirements are almost identical to the four FIPs, and are; ensuring comprehension, consciousness, control and consent. From these four usability requirements they give a very large list of 21 heuristics.

1. **Derive Use Cases**
2. **Do Object Sequence diagrams for each case-study**
3. **Apply usability principles**

Comprehension

- Comprehend how PII is handled
- Know who is processing PII and for what purpose
- Understand the limits of processing transparency
- Understand the limitations of objections to processing
- Be truly informed when giving consent to processing
- Comprehend when a contract is being formed and implications
- Understand data protection rights, and their limitations

Consciousness

- Be aware of transparency options
- Be informed when PII is processed
- Be aware of what happens to PII when retention periods expire
- Be conscious of rights to examine and modify PII
- Be aware when information may be collected automatically

Control

- Control how PII is handled
- Be able to object to processing
- Control how PII is stored
- Be able to exercise the rights to examine and correct PII

Consent

- Give informed agreement to the processing of PII
- Give explicit permission for a Controller to perform the service being contracted for
- Give specific, unambiguous consent to the processing of sensitive data
- Give special consent when information will not be editable
- Agree to the automatic collection and processing of information

Figure 8: Patrick & Kenny Framework (2003)

The Patrick & Kenny framework is also novel in that it borrows elements from software engineering techniques to structure the analysis process. This framework consists of three steps (figure 8): Derive use cases for the system, from these objects sequence diagrams are generated (a breakdown of the interactions between different system components and the user), and finally, apply a set of heuristics to these diagrams to address problems.

There are a number of interesting aspects to this framework. The first is its use of use-cases and object sequence diagrams to help the analyst break down the systems functionality. The use of these software engineering techniques means a potentially more thorough analysis is possible as they help the designer steps through the uses of the systems. The second interesting aspect of this framework is that it does not provide support for the analyst to identify potential problems in the object sequence diagrams generated, meaning that the work the analyst has done may be wasted. The only support the framework provides for this step is a list of heuristics against which solutions are to be weighted. While not specifically endorsed as a discovery mechanism, analysts are likely to employ them as such. The third interesting aspect of this framework is its exclusive focus on the user-interface and legal requirements. This framework does not directly help the analyst identify how the organization or system should go about using, storing and protecting the system.

The fifth and final framework covered in this section is the only one not primarily concerned with the usability aspects of privacy, and which does not come from the HCI literature. The *i** framework (Yu & Cysneiros, 2002) is an agent-oriented requirements engineering technique, which the authors describe as an extension of goal-oriented techniques. In agent-oriented analysis, the agents, which are the users, organizations and

systems, are the center of the analysis, which then goes on to identify the relationships, interactions and dependencies which exist between them, and the goals which motivate them.

It is important to note that though *i** was not specifically formulated to identify and address privacy goals, but rather as a way to model and analyze soft, or non-functional requirements. Privacy in *i** is dealt with as the goal of *"protecting privacy,"* which is refined from the OECD guidelines to mean the following set (Yu & Cysneiros, 2002):

- Allow Individual Participation
- Provide Openness of Purpose
- Limit Use and Disclosure of Data
- Accountability of Data Controller
- Educate Users and Private Sector
- Protect Privacy Through Transborder Data Flow Contract

The designer determines which set of these privacy goals the system needs to meet, as well as the list of other non-privacy related goals, refines these, and derives an agent-goal diagram (figure 9). These goal-decompositions can be collected and compiled into "Catalogues," and potentially re-used in other, similar situation.

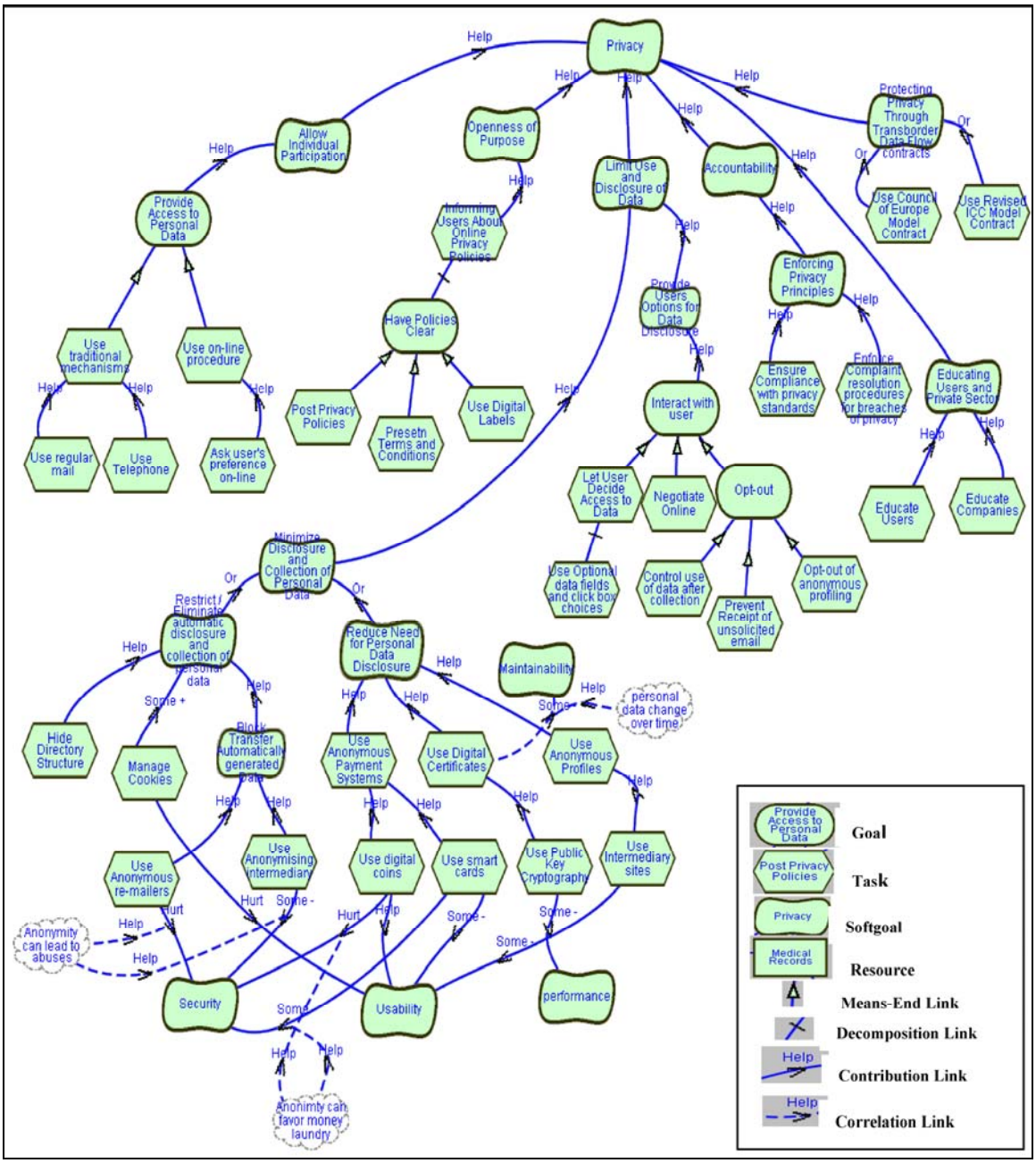


Figure 9: Modeling privacy and other requirements in *i (Yu & Cysneiros, 2002)**

Figure taken from same paper

While more structured and systematic than a heuristic evaluation, the analysis and decomposition in *i** is among the most complex in the requirement engineering literature, as may be seen in figure 9, which describes one way to refine the goals of “privacy” in the abstract. Designers are required to model goals, tasks, “soft goals” (non-functional requirements like privacy), and resources. These entities are bound together using four different types of relationships; means-end, decomposition, contributions, and correlations. Each of these entities is given a special shape in the diagram, and each type of the relationship a special arrow to denote the relationship. These shapes can be seen in the legend in figure 9.

The analyst uses these diagrams to identify potential problems and privacy requirements as well as explore different possible solutions and their implications for the overall design and functionality of the system. This includes modeling each agent’s viewpoint of the process, but is carried out without specific support or guidelines. Once the analysis is completed, a formal set of requirements can be derived and handed off to the development team.

3.2 Analysis of Design Frameworks

The five design frameworks described differ significantly on many levels which will be discussed shortly. What they do have in common is that none have been independently or extensively evaluated which makes it difficult to objectively compare their relative merits, strengths, or effectiveness. This lack of evaluation and validation may be a significant factor in the lack of adoption of these frameworks.

While one should not write off any of these frameworks, the Bellotti & Sellen framework has been around for over a decade, and *i** since 1997 (Yu, 1997). While it can

often be difficult for academic research to make an immediate and profound impact on day-to-day design practices, privacy is an area that has seen tremendous investment and interest over the last years. This should have translated to increased attention and interest in these kinds of frameworks. It is therefore important to carefully examine these frameworks to determine potential flaws or weaknesses, as well as determine and provide evidence for whether these methods work, and the kinds of problems they help identify.

One way to analyze these frameworks is to divide them into families along their theoretical foundations, those which are heuristic-based and those based on requirements analysis. The biggest difference between these two families of frameworks is the kinds of demands they put on the analyst. The Bellotti & Sellen and Langheinrich frameworks are both examples of Heuristic methods, while *i** is a good example of the later. The Patrick and Kenny framework is a hybrid of both, while the Risk models framework, while related to Heuristic evaluation, does not fit either definition.

Heuristic evaluation is a powerful technique because of its simplicity and efficiency when used by groups of analysts (Nielsen & Molich, 1991). Heuristic evaluation is simple to learn, can be applied quickly, and produces acceptable results, if analysts have sufficient support in their task. If this support is not present, designers can be prone to over-fixation and/or ignoring important problems altogether (Purcell & Gero, 1996). Nielsen & Molich used screenshots of the interfaces they wanted to evaluate to structure the process. With these screenshots it was trivial to ensure that analysts considered all aspects of the systems interface because all elements of the interface had been captured. When heuristics are used early in the design and analysis process, as is the case with privacy-aware design, this is complicated by the fact that there are few if any artifacts available. While Langheinrich does not specifically address how to structure or

guide the analysis, Bellotti & Sellen introduce a set of four questions to help the analyst structure this process, while Patrick and Kenny argue for the use of a full-blown use-case analysis.

The software and requirements engineering family of frameworks provide, and advocate the need for strong, formal methods for structuring the problem space in order to facilitate the analysis. These types of frameworks may be somewhat top-heavy, requiring more training to use, and more time spent up-front structuring the problem. Once the problem has been clearly structured, it is theoretically simpler for an analyst to identify the privacy implications of a system or design decision.

Another important difference between these two families of frameworks is their ability to support design iteration, an inevitable and desirable aspect of early design (Potts & Catledge, 1996). One of the drawbacks associated with the lack of structure in heuristic-based frameworks is that high-level changes require the analyst to re-examine the whole design with each iteration. Most structured frameworks on the other hand allow the analyst to identify how their changes ripple through the system and concentrate their attention on the parts of the design that are affected by the change.

While the frameworks classified here as heuristic-based are greatly influenced by the work of Nielsen & Molich, they deviate from the guidelines Nielsen (1994) laid down. Nielsen argues that one of the keys to the success of heuristic evaluation is keeping the number of heuristics small enough that designers can reasonably keep them in memory as they go about their task. Nielsen started off from a list of 264 heuristics and condensed this down to ten principles. While Langheinrich only lists six heuristics, the Bellotti (1997) version of the Bellotti & Sellen framework lists a total of 19 heuristics,

and Patrick & Kenny (2003) 21. It is unclear what an optimal number of heuristics might be, and whether Bellotti's or Patrick & Kenny's lists are too long to be useful or efficient.

Another important difference between these frameworks is their source for guidelines or heuristics. Bellotti & Sellen, like Hong et al., build their framework on their experiences with, or observations of, systems (in this case ubiquitous computing systems). These heuristics, like Nielsen's, are derived from real-world experience, observations of common pitfalls and mistakes. This aptly describes the frameworks by Bellotti & Sellen and Hong et al. Risk models. A different source of heuristics is a de-composition and careful study of the requirements imposed by laws and frameworks such as the FIPs and the OECD guidelines. These frameworks include those of Langheinrich (2001) and Patrick and Kenny (2003). *i** is fundamentally different from the other methods and it is not clear what group it belongs to.

Again, there are advantages and disadvantages to either of these approaches. On one hand, it is an inescapable fact that it is becoming increasingly important for systems to meet regulatory requirements. Frameworks should therefore help designers and analysts meet these requirements. On the other hand, basing heuristics on first-hand experience, or an analysis of real-world examples makes these frameworks more appealing and potentially more relevant.

While basing frameworks on real-world observations lends legitimacy and relevance to a method, it also invites questions about the size of the sample used, and the relevance of the sample to other problem domains. As an example, both Bellotti & Sellen and Hong et al. do not make claims about the applicability of their frameworks to domains other than ubiquitous computing.

Ultimately, this classification of frameworks is less important as legal frameworks and guidelines are usually derived from expert testimony, or drafted as a reaction to reported abuse. When we compare Patrick & Kenny's (2003) heuristics to those of Bellotti (1997) and Hong et al. (2004), we see that there is significant overlap and agreement between their lists. The difference between the two approaches may therefore be less important than one would think.

One important way of looking at these frameworks is determine how they integrate into and support the different stages of the design process. Privacy will only be one of many design consideration an analyst must consider, and it is therefore important that these frameworks fit into the general design process. The steps in the simplified iterative design process discussed in figure 2 (page 24) are to study users and their environment, deriver requirements, come up with design alternatives, prototype these and evaluate these, the last four of these five steps being part of an iterative process. A summary of the results of this analysis are available in table 1. An overview of Heuristic evaluation is included for comparison purposes.

All but one of the methods (Langheinrich) prescribes how the analyst should go about studying users and their environment, or at least what information they should collect. In Heuristic evaluation this was done by collecting screenshots of the interface under evaluation. In the Bellotti & Sellen framework as well as Hong et al. Risk models, this process is structured by having the analyst consider and answer a set of questions about the users and their environment. In the *i** framework, the analyst is asked to consider how to accommodate six goals into the agent-oriented model, and Patrick and Kenny ask the analyst to identify typical or key use-cases for the system.

Table 1: Privacy-Analysis Framework Overview with Design Process Contributions

Shaded areas signify processes or stages not supported by the framework

	Heuristic Evaluation (Nielsen & Molich 1991)	Bellotti & Sellen (1993)	Langheinrich (2001)	i* Framework (Yu& Cysneiros, 2002)	Patrick & Kenny (2003)	Risk Models (Hong et al 2004)
Analysis Structured By	Artifacts	Questions (4)		Goals (6)	Scenarios	Questions (11)
Problem Identification	Heuristics (10)	Questions (4)	Heuristics (6)	Goals (6)	Heuristics (21)	Questions (11)
Requirements Prescribed	10	19	6	6	21	
Evaluation Criteria	Heuristics	Heuristics	Heuristics		Heuristics	Formula Questions
Analyst Expectations	Low	Low	Low	High	High	Medium

All methods provide some form of procedure for identifying potential privacy problems, even if those are underdeveloped. In Heuristic evaluation this is done by evaluating screenshots against a short list of ten heuristics. Though the Langheinrich framework does not have a specific analysis structure, its six heuristics can be used in the same fashion. The same can be said for the twenty-one heuristics identified by Patrick & Kenny, though the method does not expressly advocate this, nor advice the analyst how this is done using the object-sequence diagrams as guides. In those frameworks where the analysis is guided by questions, (Bellotti & Sellen and Risk models these are specifically aimed at helping the analyst identify problem areas.

In addition to helping analysts discover problems, all but one of these frameworks proposes a set of requirements which a good solution should meet. In Heuristic evaluation this was the list of ten heuristics. This is a procedure which is copied by Bellotti & Sellen, Langheinrich and Patrick & Kenny, who propose nineteen, six and twenty-one heuristics respectively. The *i** framework can be said to propose its own set of six requirements in terms of the privacy goals it identifies, while the Risk model framework does not propose any requirements for a design solution.

Once problems and design requirements have been identified, designs must be generated and evaluated, preferably against other possible design solutions. This was the primary function of Heuristic evaluation, by judging whether a design or interface met a set of ten heuristics. Even though this was not the original intent, by specifying a set of heuristics the other design stages can be supported as described here. In the case of the Bellotti & Sellen, Langheinrich and Patrick & Kenny frameworks, heuristics are used in the same role as in the original Heuristic evaluation. In Risk models, Hong et al. go a different direction by introducing their formula to weigh the costs and benefits associated

with different privacy problems. This same formula may be used to evaluate the relative merits of one solution against others. In addition, Hong et al. give a set of six multi-part questions which should be used to identify a good design solution. Interestingly enough, the *i** framework does not specify a mechanism for evaluating design solutions.

Finally, these frameworks differ greatly in terms of their complexity and the amount of they require. Heuristic evaluation triumphed because it was a light-weight method requiring comparatively little time to learn compared to the much more complex, but thorough GOMS method (John & Kieras, 1996). It is important to note that simplicity does not necessarily imply an advantage when it comes to applying these methods, as conceptual simplicity may mean a lack of guidance and structure which can complicate its application. This is a question which will be explored in some depth in chapter six.

While it is possible to provide a ranking of these frameworks in terms of simplicity, this would not be a very useful thing. Instead, it is likely more productive to divide these into rough categories. Both the Bellotti & Sellen and the Langheinrich frameworks are relatively lightweight, and can be placed in the same category as Heuristic evaluation. The *i** and Patrick & Kenny frameworks are significantly more demanding, and likely in the same category as GOMS, with which they share a number of common elements. Risk models falls somewhere in-between, being more structured than the heuristic based methods, yet not as complex as the more formal methods.

In summary, what we see is a wealth of approaches, and no conclusive evidence or argument for which is most likely to help analysts identify a meaningful set of privacy problems, or derive solutions to avoid these. It is therefore important to examine both the situation most users have to deal with in terms of privacy and privacy-management solutions, and how these frameworks perform in controlled experiments.

CHAPTER 4
UNDERSTANDING PRIVACY MANAGEMENT

In order to determine how well the five frameworks discussed in chapter three address the types of privacy problems users' encounter, it is necessary to examine current privacy management interfaces. Determining how current interface design practices match users' mental models and expectations gives us a set of issues against which we should evaluate these frameworks. This will naturally not be an exhaustive investigation of the topic. Instead we do a survey of the privacy problems and practices associated with web-browsing, a task most users engage in daily and are relatively familiar with. This is a break from the domain most commonly cited for these frameworks, ubiquitous computer systems, but one which is more representative of the larger interactive systems class of systems.

In this section I discuss, briefly, three related studies into the topic of online or web-based privacy management. The goal of these studies is to offer a comparison between the needs of users, the interfaces we are offering them, and the mismatches that occur.

The first study presented is a survey of user interfaces and the metaphors and models used in these. The choice of metaphors and models is important because it influences how people think about the subject. A mismatch between the models and metaphors users have and the ones an interface is modeled on can make it more difficult for users to carry out their tasks, or make them more prone to making mistakes. Finally, different interface models make different assumptions about the level of knowledge the user has about the topic, and effort he or she is willing or able to exert

The second study is an examination of online privacy-policies, the most common and influential management and information interface for privacy in use today. Though not a very sophisticated mechanism, privacy policies are a good source of information on the privacy practices businesses employs, and which users must ultimately make decisions about. This study not only examines different usability aspects of policies and the requirements these impose on privacy management systems, but also what impact legislative efforts have had on corporate privacy practices.

The third and final study in this chapter looks at users' actual decision-making strategies and priorities in an e-commerce setting and compares those to the types of preferences stated in surveys. Most of the data we have on users' privacy practices, preferences, and concerns come from surveys. This study examines how reliable or accurate this data is in a semi-controlled setting. The goal of this study is both to determine what types of mistakes users make when managing their privacy as well as set realistic and appropriate requirements for privacy management systems.

The accounts of the second and third studies are summarized in this paper, more complete descriptions can be found in Jensen & Potts (2004) and Jensen et al. (2005) respectively.

4.1 Management Interfaces and Metaphors

One of the most ambitious visions of how privacy management should work was arguably that originally proposed as part of the P3P (The Platform for Privacy Preferences). The original vision called for users being able to specify their own privacy policies, and that the user and the website would be able to, through various software

agents and protocols, negotiate individually tailored privacy and access agreement (Cranor, 2002).

This vision not only called for a complete rethink of the privacy practices of websites, it called for a gigantic leap in the complexity and power of end-user options and control. This system would have required users to control and specify complex behaviors and conditions in order to give the system the level of control and flexibility inherent in most policies. This would have translates into a set of complex challenges in user interface and interaction design; how to enable normal users with little or no programming knowledge, and limited time and interest, to control a very complex set of variables including a large number of information categories, third parties, purposes and conditions for use, storage and disclosure. Such a framework, if done right, would have required a far more sophisticated set of user interfaces than what we see in today's privacy management systems.

The goal of allowing users to specify their own policies and allow for negotiation was eventually stricken from 1.0 specification of P3P in the interest of reaching a timely consensus on the specification document which could feasibly be implemented. It is still worthwhile discussing these ideas and goals, because they remind us of where we want to get to, what our ultimate goal and objective should be. Such a goal also helps by giving something against which we can evaluate existing user interface metaphors and ideas.

To examine what the current state of the art is in terms of web-based privacy management interfaces, I conducted a survey of such systems, or web-based systems which specifically included privacy-management interfaces (see table 3 at te end of this section for a list of systems examined). The selection criteria were simple, starting with a list of popular browsers on both PC's and Macs, relevant and still accessible tools listed

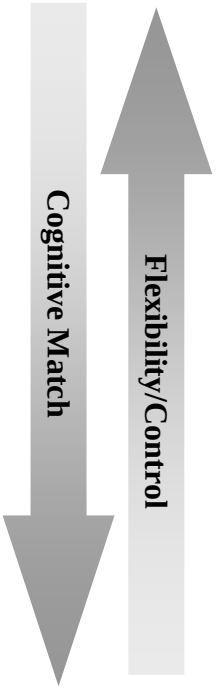
in the W3C P3P pages (http://www.w3.org/P3P/, accessed on 25/07/05), and the most popular privacy management systems according to download.com. This selection was performed on or before January 2004. Table 3 shows the results of the analysis performed on twenty-one systems. While I sought to primarily examine privacy management systems, a number of these were included in security management interfaces. The results from this survey can therefore likely be extended to cover this area as well.

More than twenty-one systems were reviewed but were not included because they lacked any significant user interface or did not offer users any meaningful options. This was especially with spy-ware scanners, proxy servers, and firewalls which only allow users to determine whether it is on or off. From this survey I found that four interface models or metaphors predominate in privacy-management; scripting interfaces, techno-centric, force/magnitude, and effect driven interfaces (see table 2 for an overview and figures 10 through 13 for examples). The interesting thing is that many applications use two or more of these models for different types of controls, or for different levels of control.

These models are largely complimentary, and can be placed along a continuum in terms of the amount of power and control they give the end user, and how closely they match the way users think about privacy and security. Surveys (Culnan & Milne, 2001, Earp et al., 2005) show that people argue about privacy in terms of effects they want to avoid, categories of information they want to protect, or information they wish to share with different groups. Occasionally, users do refer to technical details of how their information may be stolen or how they may be monitored, but their understanding of these technologies and their privacy implications tend to be weak (Jensen et al. 2005).

Table 2: User Interface Metaphors/Models for Privacy Management

	Advantage	Disadvantage	Use / Examples
Scripting	Flexibility and control – users can make their own filters and specify conditionals	Require programming experience	Proxy servers (privoxy)
Techno-centric	Balance between scripting and force/magnitude. Detailed technology, but simple options	Require knowledge and understanding of different technologies and the risks they pose	Advanced settings (IE, Mozilla)
Force / Magnitude	Simple to understand, minimal effort required from users	Artificial concept, what does medium privacy mean? Maps poorly to mental models.	Basic settings (IE, Mozilla)
Effect / Purpose	Close to users' mental models	Difficult to implement. Mapping of effect to technology unclear	PrivacyBird



Of the four main interface models, scripting interfaces are by far the simplest in that they offer the least support to the user, exposing them to all the low-level controls and features of the application. These interfaces are common in proxy servers and filtering systems, and are primarily targeted at expert or power users (see figure 10). They allow users to define their own filters and complex conditional statements, basically the full power of a scripting or programming language. This offers great flexibility and power to the small group of users who have the technical expertise to take advantage of these controls. The majority of users would likely be unable to, or not interested in managing privacy at this level of detail.

```
############################################################################
# Images:
############################################################################

# Define which file types will be treated as images, in case they get blocked later:
#
{ +handle-as-image }
/.*\.(gif|jpe?g|png|bmp|ico)$

# Known ad generators:
#
{ block-as-image }
ar.atwola.com
.ad.doubleclick.net
.ad.*.doubleclick.net
.a.yimg.com/(?:(?!/i/).)*$
.a[0-9].yimg.com/(?:(?!/i/).)*$
bs*.gsanet.com
bs*.einets.com
.qkimg.net

############################################################################
# Block these banners:
############################################################################
{ +block }

# Generic patterns:
#
ad*.
.*ads.
banner?.
count*.
/.*count(er)?\.(pl|cgi|exe|dll|asp|php[34]?)
/(?:.*/)?(publicite|werbung|rekla(ma|me|am)|annonse|maino(kset|nta|s)?)/

# Site-specific patterns (abbreviated):
#
.hitbox.com
```

Figure 10: Scripting Interface

Example taken from Privoxy 3.0.3 (http://www.privoxy.org/, accessed 25/07/05)

The Techno-centric approach is one level above that of scripting interfaces in terms of usability in that they offer users the ability to specify a set of predefined rules or actions to apply to each member of a predefined set of technologies and mechanisms. The rules and actions are typically some variant of “Disable”, “Enable” or “Prompt,” as shown in Microsoft’s Internet Explorer 6.0 (figure 11). These systems offer less flexibility than scripting interfaces; no conditionals are supported and typically offer no mechanisms to define custom technologies or actions, but are significantly easier to use in that no programming experience is needed, though they do require users to be knowledgeable of the different technologies and their risks these pose. One of the major problems with this approach is that it may not scale well. As shown in figure 11, the number of decissions users have to make grows very quickly with the complexity of the application. Microsoft’s Internet Explorer 6.0 already asks users to control 33 different technologies.

The Force/Magnitude model offers users a much simpler set of options than either the techno-centric or scripting model, and is therefore easier to use. Here users are not required to, or often even able to make decisions about specific technologies, but are rather asked to choose among a predefined set of “exposure” or “enforcement” levels (see figure 12). Users do not need to understand or know how these levels translate into individual technologies and settings, though many systems do provide some explanation.

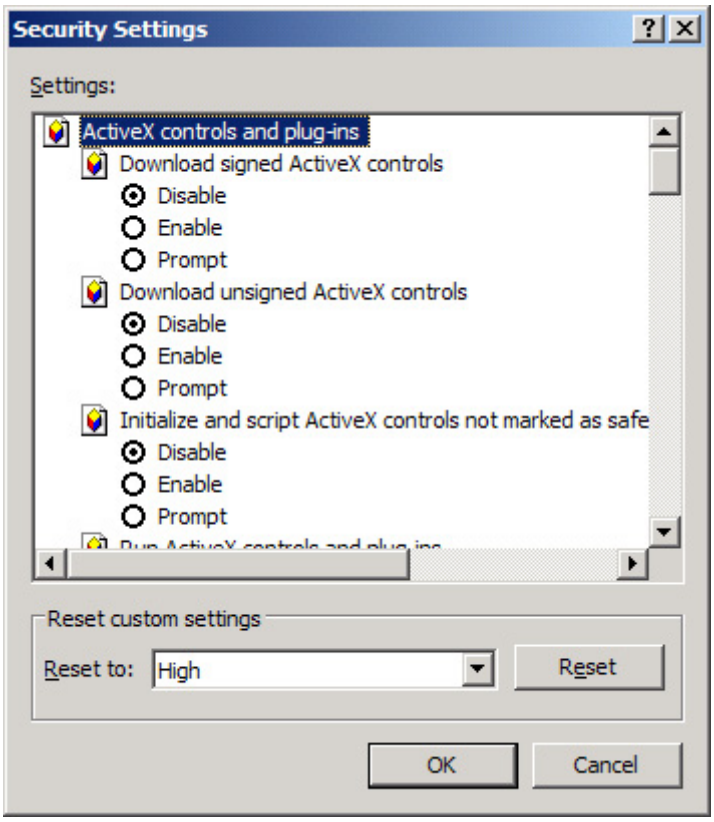


Figure 11: Techno-Centric Interface

Example taken from Microsoft's Internet Explorer 6.0 for Microsoft Windows

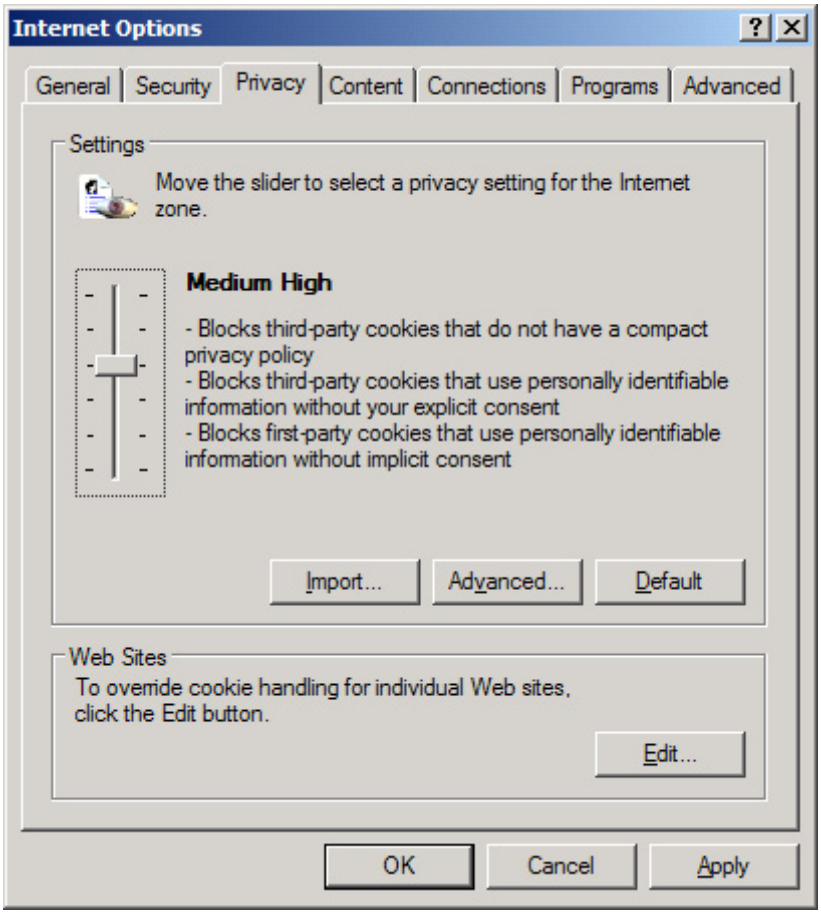


Figure 12: Force/Magnitude Interface

Example taken from Microsoft's Internet Explorer 6.0 for Microsoft Windows

These levels, while easy to use, may or may not map to users mental models. Exactly what “medium” or “Medium High” privacy means to a user or designer will likely be unclear, and mismatches are likely. This is why many of these interfaces include a brief, high-level translation of these levels. This technique is often used in tandem with techno-centric interfaces, allowing users to set rough defaults which can then be manipulated through for-instance techno-centric interfaces.

The fourth class of interfaces is relatively novel, and asks users to specify which high-level effects or information practices they consider to be acceptable or unacceptable (see figure 13). This model offers a close match to how users reason about privacy, and has primarily emerged as a response to P3P. This link to P3P is not accidental, P3P policies require policy makers to state purposes for data collection and use, which allows systems and users to make decisions about the expected benefits and risks associated with each policy element.

While these interfaces offer a better mapping to the way users think and argue about privacy, they usually require more work and effort from the user than the single setting of a force/magnitude interface. Another potential problem is that it is up to the developer to determine what combination of technologies and settings map what risks or effects. This mapping, more complicated to explain than that of the Force/Magnitude approach, may be of sufficient concern to users to avoid these interfaces. It is also unclear how this approach scales to deal with a large set of technologies or effects, or how it can deal with the discovery of new risks or exploits.

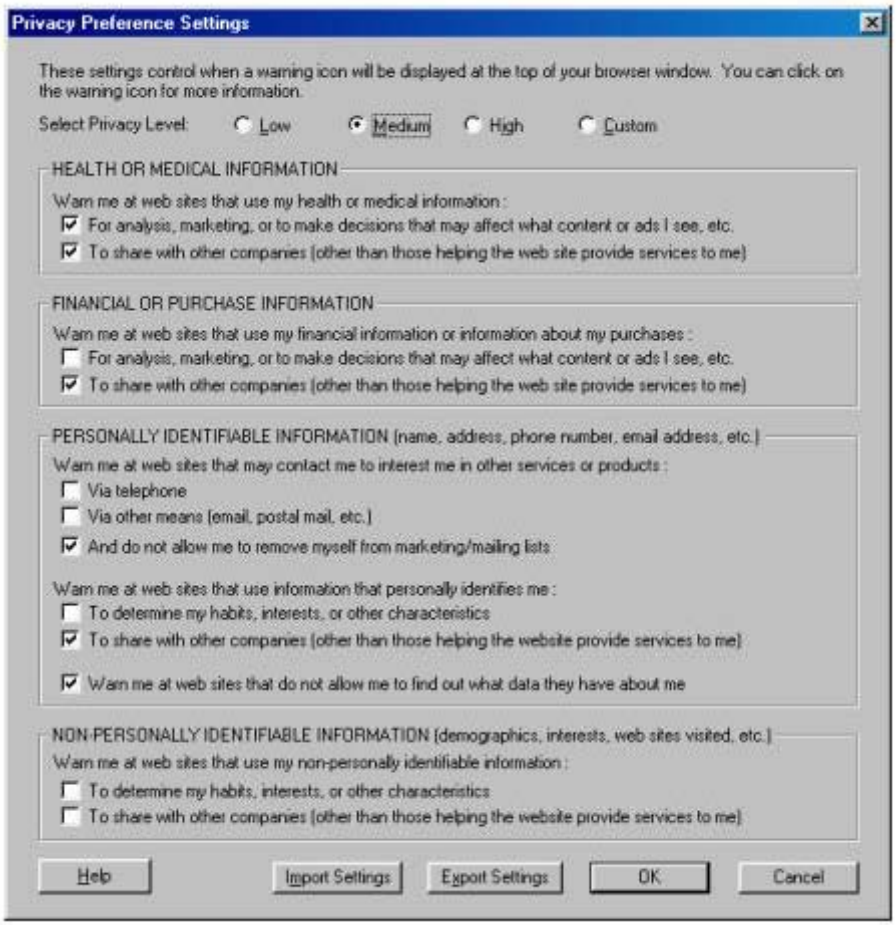


Figure 13: Effect Driven Interface

Example taken from the AT&T PrivacyBird (http://www.privacybird.com/, accessed on 25/07/05)

There are of course examples of combinations of these different approaches, as seen in table 3. The most typical combination is to use the Force/Magnitude model to set global defaults and techno-centric models to fine-tune and for set advanced settings. In this respect, no approach necessarily excludes the use of one of the others, and these models may be seen as a set of complementary approaches along a continuum of interface options.

These are of course not the only models or metaphors in use in privacy and security management interfaces. For instance, the use of zones or groups to denote territory and control degrees of access and trust are common. An example of this is the ability to define different security settings for the “Internet,” “Local intranet,” “Trusted sites,” and “Restricted sites” in Microsoft’s Internet Explorer 6.0 (see figure 14). This approach is complimentary to the four described in table 2, and is frequently used in combination with the other models. This mechanism allows users to add more detail or layers to their rules or preferences but cannot be used without one of the other models.

Another interesting approach has been to promote the use of social networks to manage privacy, as demonstrated in the Saori system (Goecks & Mynatt, 2004 and Goecks & Mynatt, 2005). While privacy is still managed at the level of cookies and other technologies which users may block or allow, these low-level decisions only have to be made by a small set of users, who hopefully will make informed decisions. The less motivated or technically savvy user can rely on the ratings of those in his social network whose judgment and motives they trust. This technique allows a relatively small set of educated and motivated users to support a large set of users, or allows a set of users to pool their resources, thereby lowering the burden on the group as a whole.

Table 3: Privacy Management Systems and Metaphor Mapping

Entries marked with an X signify extensive use, c signifies auxiliary use

System Name	System Type	Metaphor			
		Script	Technology	Magnitude	Effect
Ad Subtract Pro 2.55	Web Proxy		X		
Amit Proxy 3	Web Proxy	X			
Anonymizer Total Net Shield	Web Proxy		X	X	
AT&T Privacy Bird	Policy Analysis				X
IBM P3P Policy editor	Policy Editor		X		X
IBM Tivoli Privacy Policy Wizard	Policy Editor		X		X
Internet Junkbuster	Web Proxy	X			
JRC P3P Toolkit	Policy Editor	X	c	c	X
JRC P3P Proxy	Web Proxy	X		X	
Mcafee Security Center 2004	Firewall/Proxy		X		X
MegaProxy	Web Proxy				X
Microsoft Internet Explorer 5.2 (Mac)	Web Browser		X	c	
Microsoft Internet Explorer 6.0 (Win)	Web Browser		X	X	
Microsoft Windows Firewall	Firewall		X		
Mozilla 1.5 (Mac & Win)	Web Browser		c	X	X
Mozilla Firefox 1.0	Web Browser		X		
Muffin 0.9.3a	Web Proxy	X			
Opera 7.21	Web Browser		X		
Privoxy	Web Proxy	X			
Proximitron	Web Proxy	X	X		
Safari 1.0 (Mac)	Web Browser		X		
WebWasher Classic	Web Proxy		X		
Zone Labs ZoneAlarm Pro 5.5	Firewall/Proxy		X	X	

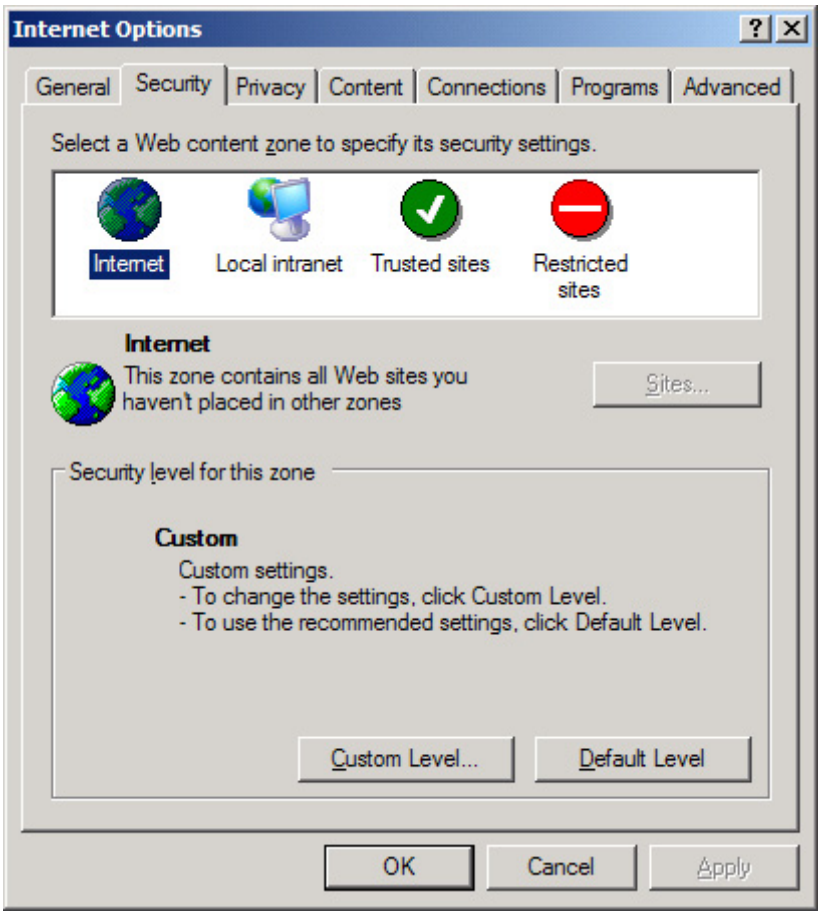


Figure 14: Group/Territory Metaphor

Example taken from Microsoft's Internet Explorer 6.0 for Microsoft Windows

Overall, a fair set of tools and models available to interface designers, offering different levels of control and abstraction to end-users. What we do have is a tradeoff between the level of control offered to users on one hand, and how well the metaphors and models match how users think and reason about privacy. The more powerful models require the user to attain a high level of technical knowledge about underlying technologies and the risks they pose, while the more naturalistic models require the abstraction of large amounts of information, and for the designer to make decisions and definitions which the user may not know about or agree with. This of course is a classic interface design problem; some models offer a good match to users thinking, some allow a lot of flexibility, but finding a model that satisfies both is typically difficult.

4.2 Privacy Policies and Practices

Privacy policies or statements are by far the most common and influential management and information interface for privacy in use today. A 2002 survey found that over 77% of the most popular web-sites posted a privacy policy, and that this number was growing rapidly (Adkinson et al., 2002). These policies are often the only source of information a user has about a company's privacy practices and they have with regards to participation and controlling information use. Privacy policies are also an important source of information for researchers, because they are our best source for understanding what practices are being embraced, and what effect legislation and public opinion are having. Finally, privacy policies are important to interface designers, because they determine the kind of vocabulary and technical terms privacy management interfaces and users will have to deal with.

The results presented in this section come from two studies which examined key usability aspects of privacy policies; their readability and how understandable they are to users, their accessibility and how easy they are to find, and the amount of burden the policies they disclose place on the users who wants to protect his or her privacy. Only the most relevant results and data from these studies will be presented here, for full results and a more in-depth discussion of methodology or implications the reader is urged to consult Jensen & Potts (2004) and Antón et al. (2004).

Another important question examined in these studies was whether the introduction of privacy legislation had improved the lot of users, or otherwise had an impact on the privacy-practices of online businesses. Most modern privacy laws, like the GLBA, introduce usability guidelines like requiring policies to be "clear and conspicuous". According to the GLBA, companies should post "a notice that is reasonably understandable and designed to call attention to the nature and significance of the information in the notice" (US, 1999 (16 C.F.R. Part 313.4(a))). In theory, rules such as these should make policies significantly easier for people to read and understand.

In the two studies, a total of 109 privacy statements from a diverse set of web-sites were studied. Specifically, three different types of web-sites were sampled; high-traffic sites, healthcare sites, and financial sites. This was done to ensure that the analysis would reflect the kinds of terms and policies most people would likely encounter in their daily lives as well as the two industries which have seen the most legislative efforts in terms of privacy in recent years; the healthcare industry under the HIPAA act (US, 1996) and the financial industry under the GLBA (US, 1999)

The first sample contained forty-seven websites from the "comScore Media Metrix Top 50 U.S. Internet Property Ranking" for August 2003[1]. The intention was to sample the full top 50, but three sites featured in this list were conglomerate sites with no common policy statement. The healthcare sample contained eighteen sites out of twenty two surveyed in an earlier study (Antón et al., 2002). By sampling the same policies as in the previous study it was possible to examine how these policies had evolved over a critical two year period which saw the introduction of major privacy legislation in the form of HIPAA (US 1996). The final sample consisted of forty policy statements from the top financial institutions in the US (by total revenue), and was sampled after the GLBA came into effect. The goal again was to see what effect legislation had on the content and usability of these privacy policies. An overview of the sites sampled and the main results are available in tables 5 through 7.

To avoid problems associated with differing legal requirements, legal standards, foreign language proficiency, poor translation, and the lack of readily available statistics on literacy rates and metrics for foreign language readability, the sites sampled were all designed for a U.S. audience, though some of the companies were foreign. While this ignores a large part of the Internet population and the sites out there, it was a necessary limitation to impose on this work.

The GLBA and a handful of other laws require companies to post "a notice that is reasonably understandable and designed to call attention to the nature and significance of the information in the notice" (US, 1999). Although the GLBA provides useful examples, they are not exhaustive and it is largely up to financial institutions to make

[1] http://www.comscore.com/press/release.asp?id=348, accessed September 2003

subjective judgments about their notices' clarity. What constitutes a clear notice hinges on the language used in that notice, and whether it is reasonable to expect the target audience to understand it. Reading and comprehension skills in turn are closely linked to educational attainment, and educational attainment with earning potential We know from the 2000 U.S. Census that 15.5% of the population over the age of 25 has less than a high-school education, and only 26.9% of the same population group has a bachelor's degree or higher (NTIA, 2002).

We know that the Internet is no longer the exclusive domain of researchers and universities; it is used by people from all walks of life. According to a recent survey, 53.9% of the U.S. population is now online, and 65.6% has access to a computer (NTIA, 2002). Literacy and education are closely linked to income and, as computers and Internet access can still be expensive, the online population has a higher than average education and literacy rate. The average education of the U.S. Internet population is 14.4 years[2] of education (the equivalent of an Associate degree or two years in college) whereas the figure for the U.S. population as a whole is 13.5 years (see Table 4). Even though adult U.S. Internet users are more educated than the average American, 28.3% of them have the equivalent of a high school education or less. As more Americans go online, the percentage of users with lower educational attainment, the most underrepresented and vulnerable population group, will inevitable grow.

Returning to the requirements specified in the GLBA, one has to determine what an appropriate threshold for "reasonably understandable" should be in this domain. Certainly, excluding more than a quarter of the population seems overly harsh a measure.

---

[2] Average assumes following years: Less than high school: 11, high school: 12, some college: 14, college: 16, postgraduate: 17.

Turning to another area where clarity and understandability have been regulated, insurance policies, we find that the most commonly applied test is whether a policy is readable to someone with the equivalent of a high-school education. If this same standard is applied online, only 3.8% of the current online population would be considered at risk.

Table 4: Education Attainment, U.S. Adult Population

Source: 2002 National Telecommunications and Information Administration report (NTIA, 2002)

Educational Level	Overall US Population			Online Population	
	Number of US residents	% of US population	% of which are online	Number of US residents Online	% of US Online Population
Less Than High School	27.5 Million	15.5 %	12.8 %	3.5 Million	3.8 %
High School GED	57.4 Million	32.4 %	39.8 %	22.8 Million	24.5 %
Some College Associates	45.4 Million	25.6 %	62.4 %	28.3 Million	30.5 %
Bachelors Degree	30.6 Million	17.7 %	80.8 %	24.7 Million	26.6 %
Beyond Bachelors	16.3 Million	9.2 %	83.7 %	13.6 Million	14.6 %

The most commonly used method for determining the readability of a text is to use a standardized statistical readability metric. This allows for an objective evaluation and simple comparison between notices. The Flesch Reading Ease Score (FRES) (Flesch, 1949) is a popular metric, used extensively to evaluate school texts and legal documents, including the readability of insurance policies in sixteen U.S. states (Antón et al. 2005). The FRES rates texts on a 100-point scale, where higher scores signify simpler texts. This score is computed by looking at the average number of syllables per word, as well as the average sentence length (Figure 15). Longer words and sentences are more difficult to read, and therefore produce a lower FRES.

The FRES can also be converted into a grade level score. The Flesch Grade Level (FGL) determines the U.S. grade-school equivalency level of a text, and is also based on the average number of syllables and sentence length. By using the FGL it is relatively simple to compare the theoretical abilities of a sample population to the theoretical demands of a text. It is important to keep in mind that both the FRES and FGL are statistical models, and though calibrated with the general U.S. population, they only offer an approximation of the complexity of a text.

A number of tools calculate the FRES automatically, including Microsoft Word, which was used in this evaluation. MS Word also calculates the FGL up to the 12th grade; for more complicated texts these scores were calculated manually using the formula in figure 15.

Flesch Reading Ease Score (FRES):

206.835 - 84.6 * (syllables/words) - 1.015 * (words/ sentences)

Flesch Grade Level (FGL):

(0.39 * words/sentences) + (11.8 * syllables/words) - 15.59

Figure 15: Flesch Reading Ease Score and Grade Level Equivalence

The most basic finding is that most policies are so complex that less than half of the US Internet population can reasonably be expected to be able to read and understand half the policies in this survey. Only 6.4% of the policies in the top-50 sample, 5.6% of policies in the healthcare sample, and 20% of the policies in the financial sample would be readable to the third of the population which only has the equivalent of a high-school education. It is interesting to note that though we cannot say anything about how the GLBA changed policies, this sample is by far the most readable, and the HIPAA sample shows no difference from the top-50 sample.

In the healthcare sample, the only sample for which we have historical data, we see a general trend towards longer and more complex policies over time. This increase comes over the same period as the introduction of HIPAA, with its requirement for clear and concise disclosure. It is reasonable to conclude that the introduction and threat of substantial penalties has led many organizations to err or the side of caution in their disclosure, increasing the number and detail of practices they disclose. This in turn has had a negative effect in terms of readability, in part because these policies are written in more legal and complicated language, and in part because the volume of information which must be processed increases. In this case, the requirement for clear and concise policies has been seen as less important than meeting the disclosure requirement and limiting what is probably seen as more serious legal liability.

While we lack data from the GLBA sample before the introduction of that legislation, it is likely that the GLBA has had a very different effect in terms of accessibility and readability compared to HIPAA. With 20% of the sample accessible to those with the equivalent of a high-school education, it is difficult to imagine the GLBA having had a detrimental effect. While 20% is far from perfect, it is significantly better than what can be expected online according to our sampling. Looking at the top of the list in terms of demands on reading ability, we find that 12.8% of the policies in the top-50 sample, 11% of the policies in the healthcare sample, and 17.5% of policies in the financial sample require the reading proficiency equivalent to that of someone with a post-graduate education.

Table 5: Length and Flesch Grade Level Equivalence: Top-50 Sample

Policy length given in number of words, FGL in number of years. Sample collected September 2003

Site Name	Length (Words)	FGL	Site Name	Length (Words)	FGL
AOL Time Warner	1101	14.87	AT&T Properties	1946	15.54
MSN -Microsoft	6222	13.18	Sony Online	3984	16.88
Yahoo! Sites	3651	12.49	Monster Property	2752	14.82
EBay	5216	13.66	iVillage	3681	16.21
Google Sites	657	11.68	Ask Jeeves	1256	14.25
Terra Lycos	5522	13.96	Weatherbug.com	3461	15.20
About - Primedia	2173	13.94	Dealtime	868	12.68
Amazon Sites	2427	14.67	Cox Enterprises	1755	17.40
Gator Network	1786	15.01	Wal-Mart	2098	12.07
Symantec	2215	12.99	United Online, Inc	4403	14.04
Excite Network	3298	15.39	News Corp. Online	2098	17.96
Viacom Online	No policy		Travelocity	403	14.53
InfoSpace Network	2033	13.76	Gannett Sites	No policy	
Walt Disney	3170	11.70	Dell	2274	11.87
CNET Networks	1723	13.26	American Greetings	3693	12.85
Real.com Network	4306	13.60	Earthlink	1788	15.17
Classmates	3542	14.57	Hewlett Packard	3301	13.44
Weather Channel	2510	14.84	New York Times	3472	12.23
Overture	1641	14.20	ORBITZ.com	3308	13.34
eUniverse Network	1099	17.14	McAfee Sites	2160	13.03
Vivendi-Universal	1729	16.02	Adobe Sites	2417	15.17
Verizon	2090	12.79	Trip Network Inc.	No policy	
EA Online	2984	14.84	Buy.com Sites	5773	13.38
Expedia Travel	4362	14.60	NFL Internet Group	2708	14.27
SBC	4693	12.97	Comcast	1158	15.48
			Average	2806.3	14.21
			Standard Deviation	1345.4	1.50

Table 6: Length and Flesch Grade Level Equivalence: Healthcare Sample

Policy length given in number of words, FGL in number of years.

		July 2001		September 2003			
	Site Name	**Length (Words)**	**FGL**	**Length (Words)**	**FLG**	**Diff length**	**Diff FGL**
Health Insurance	AETNA	806	14.20	802	14.14	-4	+0.24
	AFLAC	1930	14.98	2160	15.37	+230	+0.33
	BCBS	638	15.20	716	14.98	+78	+0.77
	CIGNA	875	10.70	1115	11.50	+240	+0.87
	EHealthInsurance	1546	15.35	2113	14.03	+567	-1.32
	Kaiser Permanente	689	14.11	4678	13.45	+3989	-0.66
	OnlineHealthPlan	1390	13.83	No publicly available policy			
Online Drugstore	CornerDrugstore	1906	12.98	No publicly available policy			
	DestinationRX	1925	13.20	1871	13.46	-54	+0.25
	Drugstore	1499	13.75	2139	14.12	+640	+0.37
	Eckerd	1340	14.02	6404	16.24	+5064	+2.22
	HealthAllies	1025	13.81	1414	14.94	+389	+1.12
	HealthCentral	1283	13.10	675	13.31	-608	+0.66
	IVillage	3382	15.89	3681	16.21	+299	+0.33
	PrescriptionOnline	753	12.69	No longer online			
	PrescriptionsByMail	1082	12.90	706	12.65	-376	+0.33
Pharmaceutical	Bayer	760	13.10	953	13.60	+193	+0.63
	Glaxo	448	12.60	396	13.19	-52	+0.67
	Lilly (Eli)	507	13.60	1014	14.76	+507	+1.15
	Novartis (Ciba)	1340	13.50	1366	13.68	+26	+.022
	Pfizer	393	12.10	331	12.39	+38	+0.57
	Pharmacia	957	13.08	Now part of Pfizer			
	Average	1203.4	13.45	1807.4	14.03	+604	+0.58
	Standard Deviation	1216.3	1.16	1613.7	1.26		

Table 7: Length and Flesch Grade Level Equivalence: Financial Sample

Policy length given in number of words, FGL in number of years. Sample collected July 2004

	Institution	Document	Length (words)	FGL
Banks	Bank of America	Overview	260	11.15
		Privacy Policy	2310	12.99
		Online Practices	1264	12.45
		Information Security	1429	11.68
		Identity Theft	523	10.42
		Accounts & Services	333	11.72
		FAQ (State: NC)	3243	11.29
	Citibank	Citigroup Promise	407	15.65
		Citi Online Data Policy	1011	15.38
		Citi MyAccounts Promise	646	14.09
		Citi MyAccounts Notice	729	15.54
		Citi Terms of Use	816	17.03
	Wachovia	Privacy Statement	2142	13.32
		Internet Privacy	2129	13.91
		Privacy Statement FAQ	2181	13.25
		Fraud Prevention	2138	11.76
		Security Statement	773	13.07
		Online Banking & Billpay	765	13.04
Insurance Companies	Allstate	Privacy Statement	2470	11.68
		Terms of Use	1583	15.86
	American Int'l Group	Privacy Policy	537	17.24
		Conditions of Use	1119	16.90
	State Farm	Privacy Principles	161	14.93
		Privacy Policy Customers	1169	12.46
		Privacy Policy Consumers	461	14.56
		Privacy and Security	291	13.14
		Privacy Policy for PHI	1221	13.50
		State Privacy Rights	205	17.08
		Privacy Policy FAQ	3542	10.78
		Terms of Use	1707	13.98
Securities Firms	Goldman Sachs	Privacy Policy	937	15.56
		Terms & Conditions of Use	1438	18.72
	Merrill Lynch	Global Privacy Pledge	1628	14.84
		Online Privacy Statement	747	12.93
		Legal Info	1816	17.27
	Morgan Stanley	Privacy Pledge	80	14.20
		US Individual Investor PP	1559	15.76
		Internet Security Policy	384	14.79
		ClientServe ISP	602	13.30
		Terms of Use	1707	17.32
		Average	1211.6	14.11
		Standard Deviation	844.4	2.09

Equally important, we find that the average policy is over 2000 words long, the equivalent of seven pages in this dissertation document. While some sites have started layering these documents, providing first a high-level summary, then providing more detailed information to interested users, this is still an overwhelming amount of information to process. This practice of layering information can also be used to hide and obscure information, as was the case with one policy in the top-50 sample, which only provided opt-out options and critical disclosures on information collection in the third layer of the policy. This having been said, the vast majority of all policies were linked directly to the site's home-page, and that these links are predictably placed at the bottom of the page or navigation menu. Though these links are sometimes obscured by the font formatting (small fonts, colors which blend in with background), the policies were generally easy to find.

In terms of usability, several common practices work against users, the two most serious ones having to do with consent. Both of these problems stem from the practice of assumed consent which websites practice. Based on the fact that a policy has been posted, sites put the burden of reading and opting out on the user. To make matters worse, the way most policies are written, consent is assumed on the first visit to the site, which means users have already consented before they follow the link to the policy. The problem stems from the practice of announcing policy changes in the policy itself. This means that most sites are requiring users to read their policy every time they visit.

These findings are important for the design of privacy management systems. We have evidence from the financial policies that notices can be written using clear and understandable terms. There is also a strong argument for the need to provide an alternative to human-readable policies. The length, complexity and amount of work

required for users to stay up to date using normal, human-readable policies makes this an unrealistic expectation. Machine readable policies could help reduce the burden on users by focusing their attention on the policy elements they should be most concerned about, which differ from the users expected practices, or which reflect a change from the last visit to the site. This in turn argues for more work and effort into the study and design of privacy-management interfaces which will provide adequate support to users.

4.3 Decision-Making and Priorities in Privacy Management

Most of the data we have about user preferences, interests and sensitivities with regards to privacy comes from user surveys. Surveys, unfortunately, are prone to different types of bias and distortion. For instance, in surveys, the way a question is worded can significantly affect the results. Another common problem is subjects' aversion to appearing ignorant, meaning people are likely to underreport reckless or inappropriate behavior. Surveys also rely on users' ability to accurately remember and summarize their behavior over days, weeks or months, which can be problematic.

Looking at some survey results, we can see clear indications of both romanticizing and/or errors of memory and synthesis. For instance, surveys routinely report large groups of users claiming to read privacy policies. For example in recent surveys 36% (Harris, 2001), 18% (Culnan-Milne, 2001), and 40% (Jupiter, 2002) of survey participants claimed it was very likely or certain they would read a privacy policy before registering with a website. The fact that the results of these surveys differ by such a large amount given a relatively large and randomly selected sample is troubling (standard deviation of 11.7%).

In addition to the inconsistencies between surveys, there is an equally important problem with these findings, and that is that they match very poorly with what we know about user behavior based on log-file analysis. Though these types of numbers are generally jealously guarded by online providers, they show that only a fraction of users (between 0.2% and 2%) ever read policies (Jensen & Potts 2004, Jensen et al. 2005), and typically only when directly asked for sensitive or very personal information.

Another problem with the data we have from surveys is that users are typically not asked to perform trade-offs or prioritize problems or desired solutions. In a situation where all solutions are available, or all problems can be avoided with no effort or cost, or in other words, no adverse consequences, users will naturally choose to do so. We know that in the face of real costs or tradeoffs have to be made, economic or in terms of time and effort, users tend to be much more selective about what they do or choose to use. This study sought to explore these sensitivities and tipping points in order to help inform designers as to the priorities of users.

This study also sought to provide the necessary data to determine whether the different analysis frameworks address the issues users care about, or are prone to miss. Given the importance of understanding user needs, actions, and priorities, it is necessary to seek more objective data on their behavior and interests. For this purpose I designed a study aimed at comparing stated to actual behavior. Naturally, behavior in controlled experiments usually does not offer a perfect match to behavior real-life behavior given that the subject usually knows he or she is being observed, and may even infer the purpose of the experiment. Regardless, controlled experiments allow us to control many potentially confounding variables, and should give a reasonable approximation of real-world behavior, or at least provide interesting contrast to the survey data.

The study consisted of four separate but interrelated tasks: (1) A basic demographic survey; (2) A survey of privacy values and attitudes; (3) A set of questions challenging users' knowledge of specific technologies and how they affect privacy; (4) An experiment presenting subjects with a series of pair-wise comparison tasks to determine the effect privacy indicators have on actual behavior. Subjects typically completed all four sections in one sitting, though they had the option to interrupt the study and return to it later. In all, subjects spent between forty-five and sixty minutes on this study. Full details of the study and the results can be found in Jensen et al. (2005).

The study was conducted online, with over 175 subjects recruited over the Internet through mailing-list announcements and postings on academic web-sites. There was a gender bias (74% or participants were male), and a strong US bias (over 95% of participants). Subjects were slightly more educated than the average Internet population. Because this was a self-selected population it is safe to assume that subjects were more educated, and more concerned, interested, or knowledgeable about privacy than the average population. This bias should not invalidate the results however, as subjects did not report higher rates of negative experiences online or with identity theft, nor did their responses to the survey questions reveal a bias compared to previous surveys.

As expected, we found a strong disparity between stated preferences and observed behavior both in terms of policy checking and in terms of subjects' knowledge of or understanding of key technologies used in the techno-centric management model (see section 4.1). While as many as 90% of subjects claimed to know what cookies were, only 14% could answer simple questions regarding the technology and why it was beneficial or harmful. For Web-bugs and P3P 35% and 22% of subjects claimed knowledge, respectively, but only 5.4% of the population could answer simple questions. This is

despite having a survey population with a higher than normal education and an interest in privacy.

To get at the more subtle trade-offs and reasoning users engage in when determining whether to trust a site, we presented subjects with a set of e-commerce scenarios. Each subject was presented with eight pairs of simulated e-commerce web-pages, one pair at a time, and asked to select which site they would prefer to buy from. Each of the pairs was designed in such a way as to be identical, except for two factors, one for each site. The goal of presenting subjects with different privacy indicators in this way was to determine which are the most influential to users' decision-making. Figure 16 shows an example of a test involving the use of the TRUSTe symbol on one hand, and credit card icons on the other. Subjects knew these were not real e-commerce sites, and that no money was being exchanged. The contents and design of the pages were in all cases similar and involved a controlled variation of twelve factors commonly cited as affecting e-commerce decision-making.

The independent variables in this experiment were: (1) Price of item; (2) visible indication of Secure Socket-Layer (SSL) encryption; (3) use of third-party cookies and P3P; (4) providing an e-mail address, (5) a telephone number, or (6) a postal address for the company; (7) the presence of a privacy seal (TRUSTe), (8) the presence of credit-card symbols (Visa, MasterCard, American Express, and Discover), and (9-12) four different types of privacy policies.

Every experiment contained some subset of these factors, meaning that a privacy policy, or contact information was not always present. The two pages being compared differed along only two of the randomly chosen factors.

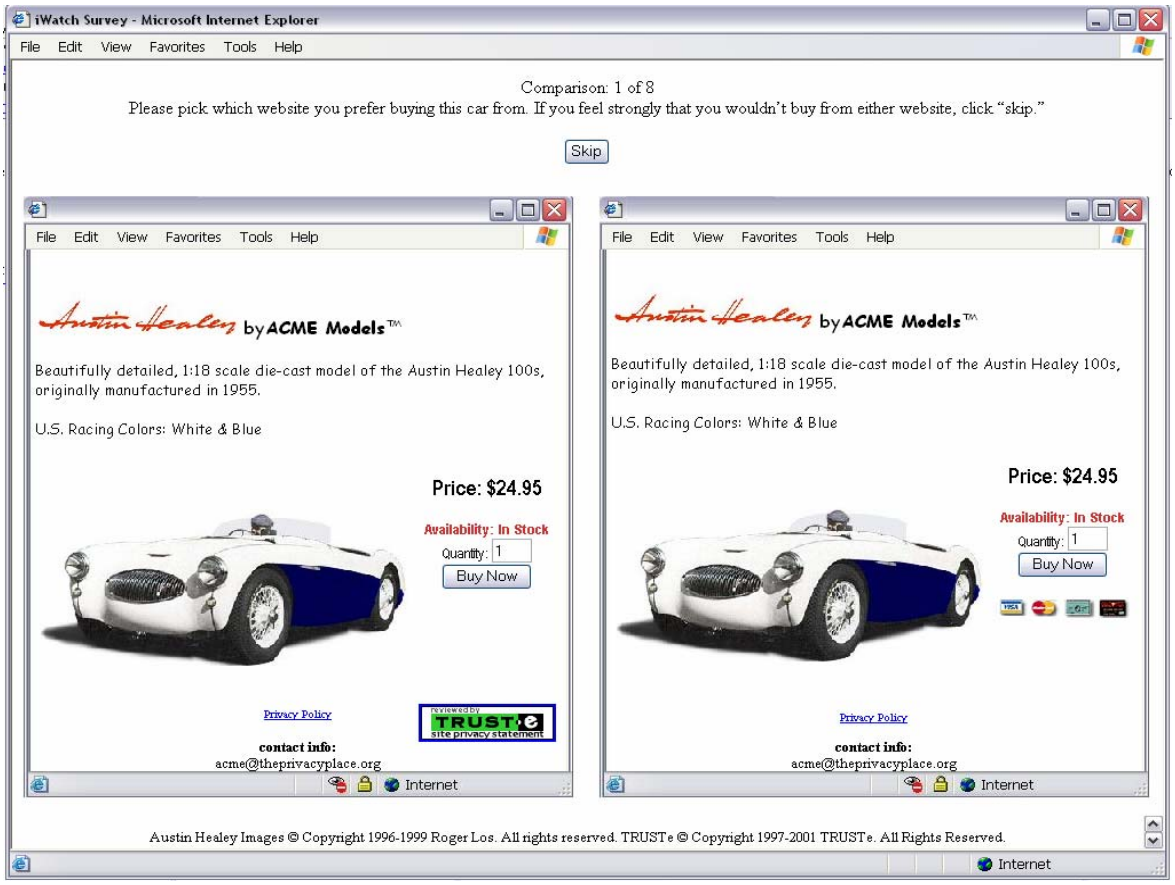


Figure 16: Forced Choice E-Commerce Experiment Comparing Privacy Indicators

Subjects must choose which of the two sites to shop from. Sites differ on two factors. In this case one has TRUSTe logo and the other has a set of credit-card icons.

The four privacy policies were written using a standard template, and varied only along two dimensions. The first dimension was whether the policy contained guarantees about the limited use of information collection and use coupled with an opt-in policy, or whether the policy informed users that they waived their privacy rights, giving the site permission to do what it wished with the information they collected. The second was whether the policy focused on the issues and practices users have indicated they care most about, or whether it focused on the issues and practices most companies are interested in disclosing, as evident by what they discuss in their privacy policies, according to Earp et al. (2005). In this way we got four policies; User-centered good, User-centered bad, Corporation-centered good, and Corporation-centered bad. In addition to keeping track of which policies were displayed, and how they affected subjects' choice, we also kept track of whether the subjects read the policy.

Not all of these variables have a bearing on users' online privacy. For instance, while the presence of contact information on the site may identify the operator and provide some means to voice complaints, it does not provide any information about the privacy practices of the site. The same goes for the presence of the credit-card icons, indicating that the site will accept payments from Vise, MasterCard, American Express, or Discovery. The overwhelming majority of e-commerce sites do accept these forms of payment, and while these credit-card companies may offer some guarantees against fraud, they do not in any way monitor or dictate how the site collects and uses personal information. The price of the item, likewise, should have no bearing on privacy. These factors were included because they have bearing on the potential cost of doing business online.

We quite frequently hear that the value of someone's privacy can easily be determined as either the going rate at one of the major data-brokers, or as the size of the discount someone must offer before they are willing to disclose all the information someone might need or want (Harris, 2003). Figures presented are typically shockingly low, often in the single dollar digits. Even if we do take a more pragmatic view, we must admit that people's actions with regard to privacy are affected by economic costs or benefits. The factors discussed above, while having no direct bearing on the privacy practices of the organization, do offer economic benefits, direct in the case of price, or indirect in the form of fraud-prevention or deterrence in case of the others.

Of the twelve factors examined, a regression analysis showed that only 11 were significant in people's decision-making, with third-party cookies and P3P proving to be insignificant. Overall, the most influential factors were "trust marks," visible indicators of what consumers take to indicate trustworthy practices, either in terms of privacy or economic. These trust marks include items such as the TRUSTe privacy seal and the presence of a telephone and contact address (physical), but also the presence of a policy link. Even in cases where subjects did not consult the privacy policy (approximately 75% of trials), the presence of a policy was highly influential in subjects' decision-making. The mere presence of a policy increased trust, regardless of what it says.

These factors influenced different people differently; therefore the factors are ranked differently depending on which subset of the population you examine. Interestingly, there were no significant gender differences in terms of privacy-practices, despite the fact that women routinely express more concerns about privacy than men. Men were slightly more knowledgeable about privacy risks and technologies than women, but women were more honest in their self-assessment. A more interesting way of

looking at the population is according to the Westin privacy segmentation (Harris, 1998), in which people are classified into one of three groups, “privacy fundamentalists”, “privacy pragmatists”, or the “privacy unconcerned”.

In the Westin privacy segmentation, “privacy fundamentalists” are defined as the roughly one quarter of the population which routinely claims to, and acts, to protect their privacy when given the option. This is considered a very important group because it is the one most willing to pay money for privacy protection tools, or to avoid services which engage in unsavory practices. The “privacy pragmatist” is the more loosely defined half to two-thirds of the population which at times acts to protect its privacy, and at times is fairly relaxed about the subject. This group comprises the bulk of online consumers, and is therefore relatively well-studied as well. The final group, the roughly quarter to ten percent of the population dubbed the “privacy unconcerned”, is characterized by their unwillingness to take action in the face of privacy threats, or their lack of concern for the subject. This segmentation has been around for a number of years, and though simplistic, is relatively stable, and popular among researchers and corporate strategists.

The Westin privacy segmentation confirmed that though we did have slightly more people at the extremes of the continuum than normal, but that our survey population was well within normal bounds for a self-selected population such as this (see table 8).

Table 8: Westin Privacy Segmentation, Historical Overview

Source: Harris Interactive (2003) report on the state of online privacy

	Harris-Westin Polls				Survey
	1999	2000	2001	2003	(Count)
Fundamentalist	25%	25%	34%	26%	34% (32)
Pragmatist	54%	63%	58%	64%	43% (40)
Unconcerned	22%	12%	8%	10%	23% (21)

When we looked at these three populations we saw that they consistently gave different answers to other privacy questions in the questionnaire, but were otherwise remarkably similar. There was no difference between the three groups in terms of the frequency and amount they claimed to spend on online purchases. There was no difference in gender, despite women's heightened concern for privacy. There was no difference in knowledge; fundamentalists were no better at answering questions about what cookies, p3p or web-bugs did, or if and why they posed dangers than the other groups. There was no indication that previous negative experiences had influenced the classification of subjects (i.e. fundamentalists were no more likely to have been victims of identity theft than others). Finally, there was no difference in how likely these groups were to actually read a privacy policy.

Table 9: Privacy Factors and Their Effect on Online Decision Making

Percentages relative to most influential factor, factors ordered by rank for general population

	All Subjects		Pragmatists		Unconcerned	
Variable	**Contrib.**	**Rank**	**Contrib.**	**Rank**	**Contrib.**	**Rank**
TRUSTe	100.0%	1	100.0%	1		
Policy-User-Good	93.5%	2	83.6%	2	100.0%	1
Policy-Corp-Good	86.2%	3			47.5%	2
Policy-Corp-Bad	74.7%	4	69.1%	5		
Contact Phone	74.6%	5	71.4%	4		
Contact Address	69.5%	6	77.7%	3		
Price Cut	62.3%	7	49.6%	9		
Policy-User-Bad	55.4%	8				
Credit Card	50.9%	9	63.1%	7		
SSL	48.8%	10	69.0%	6		
Contact Email	43.3%	11	51.8%	8		
McFadden R^2	**0.084**		**0.139**		**0.127**	

When looking at which factors influence these different groups' decision-making we finally see some significant differences. Table 9 lists the eleven significant factors, their rank in terms of importance to that groups' decision making, as well as the factors

relative contribution to the decision as a percentage of the most important factors contribution.

In the general case we see a rater complex model involving all the factors, with those identified as “trust marks” achieving the highest rankings. It is important to note the high effectiveness of privacy policies, despite the fact that they were consulted in less than a quarter of trials. The mere presence of a policy link had a strong influence on subjects’ decision-making. The Pragmatists followed with a relatively similar model, with some local reordering of factors. The unconcerned turned out to have a very simple decision-making model that despite everything accounted for almost 13% of the variance in the sample.

One interesting finding was that we were unable to find a statistically significant model for the “privacy fundamentalists”. None of the twelve factors turned out to be significant in the groups’ decision-making. While we had expected this group to have different priorities or strategies, this was an unexpected result. This result does not mean that subjects in this group did not consider any of these factors, but rather that subjects in this group did not form a cohesive enough group to allow us to generalize their behavior. In other words, while we found strong evidence for the fact that there are such things as “privacy pragmatists” and the “privacy unconcerned”, the “privacy fundamentalists” seem to be composed of several subgroups rather than act as a single coherent group. This is a very strong argument for the need to learn more about this very influential group.

The findings of this study have clear implications for the design of future interfaces, the way we interpret survey findings and our understanding of user goals, behaviors, and expectations when it comes to privacy. These findings also have clear

implications for some of the most popular interface metaphors (the techno-centric especially), and their suitability to the task. People simply do not have enough knowledge, or even the right mental models to reason about these technologies. We also see that subjects rarely consult policies, but are greatly influenced by their presence, and those of other “trust marks”. This is an indication that users want simple and clear privacy indicators rather than more complex information.

CHAPTER 5
STRUCTURED ANALYSIS OF PRIVACY (STRAP)

In chapter three the most influential frameworks for privacy-aware design were reviewed. As part of this review I analyzed their theoretical foundations, their assumptions, and their strengths as well as their weaknesses. This survey showed that no framework is perfect, and all frameworks have a set of limitations and potential flaws. As long as those are well known and understood, there is no reason why the framework may not continue to be used successfully. The five frameworks presented are fundamentally very different, building on different principles, and offering different solutions to the problem of designing for privacy. Because they are so fundamentally different, and because no independent and extensive evaluation has been performed on these frameworks, it is important that we look very carefully at their relative merits and problems.

The five frameworks could be divided into families along a set of different criteria, the first of which is the amount of structure or support they provide the analyst in his or her task of discovering problems. In this case we saw that there were two major families of frameworks, those which relied on questions and heuristics to guide the analysis, and those which relied on more elaborate requirements engineering techniques. While the first approach is simpler to teach and apply, the second has the potential for a more rigorous and thorough analysis process, though at a potentially high cost in terms of training and time on task. One important question to answer is which of these fundamentally different approaches is more effective and efficient in this space, and which of these methods are better at identifying and dealing with what kinds of privacy

problems. To answer this and other questions, as outlined in chapter three, it is important to look at these frameworks more carefully.

There are a number of different ways of examining these questions as well as those listed in chapter three. In this dissertation a two-pronged approach is employed to examining these issues; by conducting a number of design experiments in which groups of analysts are asked to use different methods to analyze the same system, and by developing and testing a new framework, STRAP, based on the theoretical analysis in chapter three, as well as the analysis of the problem domain in chapter four. By including this custom framework in the design and analysis experiments, it is possible to gain deeper insight into the reasons why different frameworks excel or fail at supporting different tasks.

The overriding goal behind formulating STRAP was to address some of the critiques and potential shortcomings of current privacy analysis frameworks, while building on the advantages and strengths identified. By doing so, I seek to demonstrate the validity of the theoretical analysis performed in chapter three as well as provide a tool which will allow analysts to perform more in-depth and efficient forms of analysis. STRAP is also meant to allow us to test whether it is possible to successfully combine these different aspects of frameworks, based on such different principles, into a single, coherent, and efficient design framework.

More specifically, STRAP is designed to strike a balance between the robustness of the requirement engineering based techniques, while retaining the flexibility and ease of use of the heuristics-based approaches. STRAP combines these two methods, goal-oriented analysis and heuristic evaluation, in an effort to improve the quality and effectiveness of the analysis while keeping the costs down.

STRAP is primarily aimed at supporting designers and analysts in dealing with privacy in the early phases of design when functionality is still being decided and fundamental design decisions can still be changed or influenced. STRAP is also specifically aimed at dealing with the HCI problems associated with privacy awareness and management. This means that the heuristics used are specifically targeted to deal with these issues. This does not mean that STRAP will not be useful in analyzing and discovering other types of privacy problems, though this is a question which will have to be determined experimentally.

In this chapter I will give a detailed descriptions of STRAP, as well as the underlying decisions and foundations on which this framework builds. I then present a detailed example of how STRAP is applied by analyzing a hypothetical web-browser and deriving a set of requirements for a privacy-aware browser. Finally, I describe the process of going from requirements to the implementation of iWatch, an experimental privacy awareness and management tool.

5.1 Privacy-Aware Design with STRAP

STRAP is a method in five steps.

- Goal-Oriented Analysis
- Vulnerability Analysis
- Goal Refinement and Design
- Design Evaluation
- Iteration

STRAP is a hybrid framework in that it uses both a structured requirements gathering techniques as well as more flexible and light-weight heuristic methods.

Specifically, it uses a goal-oriented analysis technique to structure the problem space, ensuring a thorough and balanced analysis, where light-weight and efficient heuristics are used to identify potential privacy problems, suggest solutions or design refinements, and finally help the analysts evaluate these refinements. Because this framework is based around a structured goal-oriented analysis, iteration is also supported. Evaluation Each of these phases is discussed in the following sections.

5.1.1 Goal-Oriented Analysis in STRAP

Goal-oriented analysis is a technique typically associated with requirements engineering (Dardenne et al., 1993). It is used to elicit system requirements, and structure how analysts and designers think about a system early in the design process. In goal-oriented analysis, the analyst identifies the systems goals from a users' perspective, by studying users, the way they interact with each other, the systems, artifacts, or the environment. From these observations, requirements can be identified, either directly or by deriving scenarios and use cases which illustrate typical as well as critical functions or tasks. In this role, goal-oriented analysis can be used to structure and analyze complex sets of observational or unstructured data.

After identifying key goals and functions, these are broken down into increasingly simple component sub-goals, much in the same way as in the GOMS method (John & Kieras, 1996). This process continues until the sub-goals are trivially simple, and further breakdown is counter-productive. This is a somewhat weak definition, as different analysts will have different thresholds for what is self-evident and what is not. This can of course also influence the quality of the analysis and whether certain types of vulnerabilities are discovered or not. An alternative definition of how far a set of goals

need to be decomposed is to that all leaf-goals must be assignable to a single actor. In other words, all leaf-goals must be so simple that a single entity or system can fulfill them completely. Until the analysis reaches this level, implementation details and necessary decisions will likely be hidden or lost in the analysis.

Goal-oriented analysis is a familiar technique to most designers and developers, bearing close resemblance to other techniques such as hierarchical task analysis (Shepherd, 1985), or object-oriented modeling and design (Rumbaugh et al., 1991), and should therefore be more readily adopted than many other, and often more formal requirements engineering methods. Furthermore, goals, in contrast to requirements or tasks, are approximations of system properties or functionality, and may therefore be more appropriate for early modeling and analysis. Goals are idealizations that are not necessarily fully achievable in the "real world" where many factors which affect the design may be outside the control of the designer. These factors may include physical properties of the environment that may be predicted or influenced but not determined absolutely, other systems that may not behave as assumed, and the implementation of some exception-prone aspects of the proposed system itself.

This ability to deal with ambiguity and conflict makes goal-oriented analysis ideal for the analysis of privacy and security which often includes actors with their own goals which may be antagonistic to those embodied by the system. In requirements engineering, it is often assumed that the refinement and allocation of goals is a rational and beneficial activity, but nothing in the goal-refinement approach requires this, and recent research has turned to the analysis of adversarial goals in security applications (van Lamsweerde, 2004). Indeed, in a multi-actor domain, such as e-commerce or groupware systems, there is ample opportunity for conflicts or tradeoffs among the goals

of users. When antagonistic goals make the fulfillment of the original goal impossible, this is referred to as an "obstacle." Such an obstacle is an "anti-goal," a set of events which make it impossible for the goal to be satisfied. In the domain of privacy, we refer to obstacles as vulnerabilities, because they indicate a potential privacy risk rather than the certainty that a goal will be blocked.

STRAP builds on a family of goal-oriented analysis techniques called GBRAM (Antón 1996, Antón & Potts, 1998) and its extension to account for obstacles (Potts, 1999). Using these techniques, an analyst derives a tree, a set of interconnected goals which can be used as a visual artifact to guide the application of heuristics much in the way Nielsen & Molich (1990) used screenshots. The exact representation of these goals is not crucial to the analysis or the method, what is essential is that there be a thorough analysis of the system, and that at the end of the process there be some visual representation to anchor the rest of the analysis.

To anchor the discussion, this dissertation will present and use a particular graphical representation. In this representation, goals and sub-goals are drawn as circles, the top decomposed into lower level circles, as denoted by the arrows. Actors responsible for goals (user roles, system components, environment entities, etc.) are typically identified by color-coding the nodes. Arches along the paths denote an 'or' operator, the absence of such an indicator usually meaning that the sub-goals are all mutually compatible or interdependent. While the left-right ordering of the child nodes do not necessarily denote order of operation, we have attempted to accommodate that reading.

Goals sometimes refer to each other up and down the goal-tree, or recursively. In cases where this happens one can short-circuit the analysis. Because these goals refer to the same set of functions that have already been analyzed, we know no new

vulnerabilities will be introduced by calling the function again. The analyst can therefore simply note the recursion and move on. Finally, each leaf-node in this graph is assigned to an agent in the domain whose ultimate responsibility it is to ensure that the goal is met. While not always necessary, this can help determine which components are most relevant to modify, or what stakeholders may be affected should goals need to be modified.

Vulnerabilities are drawn as clouds along the paths of the graph with callouts describing them. This placement is meant to signify that the goal can be blocked by the vulnerability. The descriptions in the callouts should contain the type of vulnerability and a brief description of the problem. These callouts are intended to serve as placeholders and visual reminders, ensuring that the context is preserved. For more in-depth analysis of the problem, the analyst should keep external notes and scenarios for the potential problems.

This is of course a very abstract and high-level overview of how this method works in practice, and a poor guide to those wishing to learn how to apply it. For this reason, a comprehensive, step by step example of how this goal-decomposition is done in practice has been provided in chapter 5.2.

5.1.2 Vulnerability Analysis in STRAP

Using the goal-tree derived in the previous step as a blueprint or checklist of the functionality to consider an analyst can set out to identify potential privacy vulnerabilities. Both heuristic-based and requirements engineering analysis methods commonly rely on a small set of analytical questions to identify potential problems (see figure 6 for Bellotti & Sellen (1993) or figure 7 for Hong et al (2004) for examples). For STRAP, these questions were derived from Bellotti & Sellen (1993), as well as a review

of the information life-cycle, potential sources of information misuse, and the kinds of questions users have about companies information practices (Earp et al. 2005). For each goal and sub-goal, the designer should ask the following questions to determine the information capture and use:

- What information is captured, accessed, processed, transmitted, received, or stored in meeting this goal?
- How is the information captured, accessed, processed, transmitted, received, or stored?
- Who are the actors involved in the capture, access, processing, transmission, reception, or storage?
- What knowledge is derived from this information?
- What is done with the information and meta-information after the goal is met?

If information is captured, accessed, processed, transmitted, received, or stored, this may present a privacy vulnerability. Vulnerabilities are marked in the goal-tree as clouds over the path to the goal. By placing these vulnerabilities in the diagram context information is preserved; we know what goals these problems are associated with and what actors are involved. This helps give the designers, programmers, and users a shared vocabulary for discussing the system. Furthermore, when goals are finally translated into requirements, functions, and eventually system components, we can trace how these vulnerabilities are addressed, or will affect different system components.

Heuristics, as in Nielsen's framework, can play an important role in the identification of potential problems because they list common flaws or requirements for a good design. In the case of STRAP, a list of eleven heuristics has been defined, and though they are discussed in-depth in the evaluation section, an analyst should consider

these issues in the analysis and design phases as well. Figure 17 contains an overview of these heuristics.

At this stage of design, it is important to note that most privacy vulnerabilities are only potential privacy problems associated with the design rather than serious flaws. Because few if any technical or implementation details should be set yet, it is still possible to avoid these problems through the use of appropriate technology or implementation decisions. For instance, the transmission of information over a network need not be a problem, if adequate safeguards are put in place, such as encrypting the information. It is therefore important that the analyst keep an open mind, think of the worst-case situation, and document every potential flaw, even if the solution is self-evident. This is a crucial step because it helps the designer document any hidden assumptions which must then be communicated to the developers and users, and kept in mind as the system evolves.

Once vulnerabilities have been identified, we look for common causes and duplicate vulnerabilities. These often occur when one set of goals collect, transmit or store data needed to meet a different set of goals. The vulnerability will then appear in two or more places on the goal-tree, with different contexts. If the vulnerability associated with the transmission of data is addressed through the encryption of that data, then the vulnerability associated with receiving the information will also be addressed. These duplicates are important to identify, not simply because it simplifies the analysts job later on, reducing the number of vulnerabilities to address, but because it affects the cost-benefit analysis which should be a part of the decision about which issues to address or not.

As both Butler & Fischbeck (2002) and Hong et al (2004) argue, it is important to understand that it is often impossible or undesirable to address all vulnerabilities in a system. In some cases the implementation or technology cost would be prohibitive, the fix could seriously undermine the system's utility, or the fix could introduce other, more serious problems. In some cases the risks associated with a vulnerability may simply be too small to warrant any action. Vulnerabilities resulting from dependencies on other systems, such as the operating system, are especially difficult to remedy. In such cases, designers may be limited to noting the problem and warning users and administrators.

5.1.3 Goal Refinement and Design in STRAP

The goal refinement stage is where the analyst should start to think about how different design solutions, or different compromises will affect both the systems overall functionality, performance, and users' privacy. As in any design process, several competing solutions should be derived and evaluated, where possible by independent design teams. These competing designs should then be evaluated to identify the most successful design (or design elements) before prototypes are implemented and evaluated.

The first step in the refinement process is to determine which vulnerabilities can be eliminated and which must be mitigated through the redefinition or elimination of goals. This is the stage where the analysts must negotiate with the different stake-holders what the functionality and system specifications need to be. Vulnerabilities may be eliminated or mitigated by removing the originating goals, by changing the goals of the system and therefore the information requirements to meet those goals, or by dictating implementation requirements.

For example, a goal which requires the storage of personal data in a database for some customization introduces a privacy vulnerability by potential exposing this information to misappropriation or misuse. This vulnerability can be eliminated in a number of ways. Either the goal of customizing the content to the user can be removed, at a cost (loss of functionality) to the user, thus eliminating the need to store the data. The goal can be modified so that the vulnerability is eliminated (by only storing harmless data), or mitigated (limiting access and use, or anonymizing the data). Alternatively, the analyst can dictate implementation requirements which may mitigate the vulnerability (database must be encrypted).

Mitigation, though less desirable than elimination, is an important strategy because it gives the designer more options for striking a balance between functionality and risk to users. It may often be the case that the costs of eliminating a set of vulnerabilities would be to severely limit the functionality of the application, and thereby its value both to the owners and the users. The principal strategy to follow in these cases is to try to reduce the risk or cost associated with a potential privacy violation by limiting the information stored, or installing safeguards which will make it less likely for data to be misused.

Mitigation strategies are also important for those vulnerabilities where users are likely to differ in sensitivities and preferences. Palen and Dourish (2003) describe privacy management as a highly individual and dynamic process, where decisions are highly dependent on personal preferences, expectations and context. In most non-trivial systems it will be impossible to come up with universally acceptable designs or tradeoffs. Instead we shift to mitigation strategies involving the user in the decision-making process. An example of such a mitigation strategy is to inform the user when information

is being captured and the risks they may run by agreeing to such a service. While the risk is not reduced, the cost is somewhat reduced by making the user aware of the potential danger and allowing them to opt out.

Though design is an inherently creative process and there are no formulas for coming up with design solutions, this process can be supported or guided by looking at the list of heuristics used to evaluate solutions. For a list of heuristics proposed for STRAP, see figure 17. This set of heuristics in effect offers a concise list of best practices or requirements for what a good solution would need to offer, and can therefore serve as design patterns.

There are also a number of promising strategies to follow in the case of vulnerability mitigation. As described earlier, the simplest mitigation strategy is to engage the user, letting them make their own decisions by informing them of what the system is doing, and whether they consent or decline. Today, this is usually a rather poor strategy because users tend to be inundated with irrelevant or complex information, not given the right options to make or enforce decisions, or because they are asked to make decisions without adequate contextual information. Users do not appreciate the risks they face when faced with an all encompassing disclaimer when they start using a service because they do not see the implications of their decisions, or when and for what reasons they may be put at risk. Through STRAP, more effective notices may be given as the analysis preserves the context of the vulnerability, allowing the designer to give just-in-time and context-rich notices and options.

Other ways of improving this strategy is to make the decision-making less demanding on users. While privacy management is a dynamic decision making process, high-level polices and plans can serve as the basis for basic risk assessment. These can

serve as the basis for more advanced and less intrusive UI approaches such as mixed-initiative systems (Horowitz, 1999) or ramping interfaces (Rhodes, 2000). These techniques minimize the distraction to the user by determining what the user needs or wants to know, and disclosing more or less information as needed.

In a mixed-initiative approach, a high-level policy is evaluated against the risks associated with the disclosure or withholding of information. The cost of distracting the user also factors into the calculation. The result is an expected utility for each of four possible actions: correct automatic disclosure, incorrect automatic disclosure, correct withholding, and incorrect information withholding. The utility values dictate what action to follow, or when the user should be prompted to make the decision (utilities too close or low to discriminate). Over time, user actions can inform the model, resulting in a more detailed, flexible, and individualized model, progressively becoming less invasive.

These are only some of the possible strategies for mitigating vulnerabilities and involving users. Other techniques, such as attention-based interaction or peripheral awareness interfaces could also be employed to minimize cost of involving users in the decision-making. Social solutions such as collaborative filtering could also be used to inform decision-making for privacy, as demonstrated by Goecks & Mynatt (2004). The application domain, the knowledge and motivation of the user population as well as other constraints will dictate which techniques are feasible, or desirable in any given situation.

5.1.4 Design Evaluation in STRAP

There are a number of different criteria against which designs must be evaluated, as is the case even when privacy is not a concern. A system should have value, it should address a need, it should be usable, and it should be safe. These are only some of the

common criteria against which designs are evaluated. When it comes to evaluating different alternatives for addressing privacy vulnerabilities, the first consideration should be to look at the risks eliminated and the costs associated with the different design solutions. A designer should seek to find the optimal tradeoff between usefulness, implementation costs, and privacy protection. How these different concerns are weighted is highly dependent on the application and the context for use. For example, designers of health-care applications should be more sensitive to privacy problems than development costs, whereas small companies working on non-commercial web-sites might have very different priorities.

This need to make tradeoffs has been discussed in length by Butler & Fischbeck (2002) and Hong et al. (2004), and while one can argue about the merits or even feasibility of attaching dollar figures and specific probabilities to individual vulnerabilities, one must accept the fact that not all vulnerabilities may be worth addressing, or that in the face of finite resources, a designer will most likely have to prioritize or compromise. It is therefore important for designers and analysts to consider the costs and risks associated with vulnerabilities and design solutions and somehow rank these.

An important part of this evaluation of course, is how adequately the proposed design change addresses the potential vulnerability, or how desirable the design change is. For this, heuristics are an appealing evaluation tool, combining ease of use with efficiency and low cost of entry. In the methods reviewed in chapter three, the use of heuristics dominated the evaluation step. The question then becomes what these heuristics should be based on, and what would be a suitable set of heuristics for tackling this problem domain.

As discussed in chapter three, there are two families of heuristics used in privacy-aware frameworks, those based on experience with systems and their deployments (Bellotti & Sellen, 1993, and Hong et al, 2004), and those based on the analysis of legal frameworks (Langheinrich 2001, Patrick & Kenny 2003). While heuristics based on observations are appealing because of their validity, their applicability may be limited, depending on how generalizable a sample they are derived from. Heuristics based on legal frameworks may be less domain specific, but they may have less relevance to the kinds of problems users and designers encounter.

In STRAP, we again use a hybrid approach, combining an outline based on the analysis of legal requirements with detailed heuristics based on an observation of the kinds of problems users encounter with policies and managing their online privacy. The heuristics for STRAP were derived from an analysis of the OECD guidelines (OECD, 1980) and FIPs (FTC, 2000), augmented and filtered by the principles in Nielsen (1997), Bellotti & Sellen's (1993), and Hong et al.'s (2004), the practices and problems observed in privacy policies (Jensen & Potts, 2004), and a study of privacy dependent decision-making strategies and pitfalls (Jensen et al. 2005). Most of this data has been discussed earlier in this thesis. By building on an analysis of these rather general frameworks as well as common disclosure practices and pitfalls, I hoped to derive a set of heuristics relevant to the strategy advocated by STRAP, that of engaging and empowering users to manage their own privacy.

By looking at all these sources one can derive a lengthy list of heuristics and best practices. However, as Nielsen pointed out, keeping the list of heuristics small is an important factor in achieving efficiency (Nielsen, 1994). While no exact figures exist on what an ideal number of heuristics are, or whether this is a domain-specific question, we

sought to keep the number of heuristics used in STRAP close to ten, which is what Nielsen used. Finally, after much debate and consideration, a list of eleven heuristics was proposed. These heuristics are shown in figure 17.

The process of selecting or formulating these heuristics was relatively simple. Starting with an analysis of legal frameworks, the FIPs were chosen as a simple framework from which to start. The four FIP principles were broken down into more detailed requirements, and further augmented with heuristics and requirements from other frameworks. This list was then carefully examined and similar heuristics grouped and duplicates or overly-similar entries eliminated. The list was then condensed to no more than one dozen heuristics based on overlap with other heuristics or how critical a heuristic was considered. These heuristics were then circulated among colleagues and critiqued. When new heuristics were suggested as part of this process they were added to the list and the list was again condensed.

Though the FIPs categorization of the heuristics was an artifact of how this process started, this organization was later found to be useful in identifying design solutions. This is not a very rigorous classification; some of these heuristics could easily fit under more than one FIPs category, and should only be seen as a rough guide.

As in Nielsen's heuristic evaluation, these heuristics can be used to identify problems and flaws, suggest design requirements and constraints, as well as evaluate proposed design solutions. In evaluating, or selecting a set of design solutions to address an optimal number of vulnerabilities, the designer should look for solutions which meet the properties identified by this set of heuristics.

1. **Notice/awareness**
   a. **Available, Accessible and Clear** – *Make information about the systems activities be always available to users in a way that is simple to access and understand*
   b. **Correct, Complete and Consistent** – *Ensure that the above disclosures are complete, correct and consistent in order for users to make informed decisions*
   c. **Presented in context** – *Relevant information should be presented for each transaction to minimize memory load and ensure users are aware of the consequence of their actions*
   d. **Not overburdening** – *Disclosure must take into consideration human limitations in memory, ability, and interest. Provide succinct and relevant information*
2. **Choice/Consent**
   a. **Meaningful options** – *Users need to be given real options rather than opt-in/opt-out when possible to avoid coercion and maximize benefits*
   b. **Appropriate defaults** – *The systems default settings should reflect the (majority of) users concerns and expectations with regards to protecting their privacy*
   c. **Explicit consent** – *A system should avoid assuming consent whenever possible.*
3. **Integrity/Security**
   a. **Awareness of security mechanism**s – *Users should be provided with enough information to judge the security of the system and their information*
   b. **Transparency of transactions** – *Systems should provide transparency of transactions and data use to build user confidence and trust*
4. **Enforcement/Redress**
   a. **Access to own records** – *Users should have access to all information the system has collected about them, regardless of source*
   b. **Ability to revoke consent** – *Consent should be retractable*

Figure 17: STRAP Heuristics, organized by FIPs category

5.1.5 Iteration in STRAP

Most design processes are naturally iterative and design methods and tools should therefore support this practice. Iteration is important for two reasons, the first being that in redefining goals to eliminate or mitigate vulnerabilities, new sets of vulnerabilities may be introduced. This is often the case when awareness and management support is added, as this often requires the system to keep a closer tab on the user, their activities, and what they have disclosed to different systems. The second reason is that privacy will typically only be one of many factors to be considered in the design process. This typically means that it will be necessary to go through a number of design iterations and alternatives before a satisfactory balance is found.

Some methods are more suited to support this iteration process than others. When using a heuristic method in the manner described by Nielsen & Molich (1990), which means centered on the interface artifacts, iteration can be accommodated relatively efficiently by only re-examining those artifacts which have undergone some change. When heuristic methods are used to evaluate less structured problems or hypothetical systems, as is the case here, there are no artifacts to constrain the analysis, nor are there other indicators which would help the designer determine how a change may have rippled through a system. An analyst would therefore likely have to reexamine the whole system for each design iteration to make sure nothing is missed (Potts & Catledge, 1996).

Because STRAP keeps track of the system as expressed in the goal-tree, changes and their effects can easily be tracked. An analyst can limit his reexamination to examining how the new or altered goals will affect the other goals in the goal-tree. The analyst does not need to derive an entirely new goal-tree, thereby lowering the cost of iteration substantially. This goal-tree can also be used to document how and why design

decisions were made, and the hidden assumptions that were made. As the system evolves, this document evolves, and old assumptions can be checked for validity as needed.

5.1.6 Discussion

This has been a whirl-wind tour of STRAP as a method, and a brief discussion of the decisions and influences which shaped the development of it. The overarching goal was to formulate a framework which integrated what theoretically should be the strengths of previous frameworks, while striking a balance between rigor and ease of use.

The use and selection of analysis question and heuristics was heavily influenced by the analysis of legal frameworks (chapter 2.4), existing design frameworks (chapter 3), and the kinds of problems and challenges users encounter in managing their online privacy (chapter 4). While most of the analysis of problems was focused on web-based problems, the heuristics selected were intended to be as generalizable as possible. This being said, there was a bias towards the types of heuristics which would identify awareness and control-type problems, prime targets for the application of innovative interaction techniques.

These types of biases are inevitable. When the number of heuristics must be limited, one must set priorities. When there is an absence of quantitative data on the frequency and severity of incidents associated with each heuristic, there is no way to objectively select which heuristics should be included and which should be cut from the list. While this is an important limitation to be aware of when using STRAP or any heuristic-based framework, it does not invalidate the method. One should instead be careful about reading too much into statistics discussing the prevalence of one type of vulnerability compared to others as this may be an artifact of heuristic bias.

Table 10: Privacy-Analysis Framework Overview with STRAP

Shaded areas signify processes or stages not supported by the framework

	Bellotti & Sellen (1993)	Langheinrich (2001)	i* Framework (Yu& Cysneiros, 2002)	Patrick & Kenny (2003)	Risk Models (Hong et al 2004)	STRAP
Analysis Structured By	Questions (4)		Goals (6)	Scenarios	Questions (11)	Goal-oriented analysis
Problem Identification	Questions (4)	Heuristics (6)	Goals (6)	Heuristics (21)	Questions (11)	Questions (5)
Requirements Prescribed	19	6	6	21		11
Evaluation Criteria	Heuristics	Heuristics		Heuristics	Formula Questions	Heuristics Value
Analyst Expectations	Low	Low	High	High	Medium	Medium

It is important to note that by the nature of the goal-oriented analysis technique employed, a different analyst would have likely derived a goal-tree which looked different and identified different goals. This is an artifact of different perspectives on organization, or how the scenarios or use cases are organized and analyzed. The important thing to note is that if the analysis was done correctly, the two goal-trees, though superficially different from each other, should prove to be functionally equivalent. For this same reason it is important to note that it will not be possible to compare raw numbers of vulnerabilities, both within and between frameworks. These counts should first be normalized; duplicate vulnerabilities need to be removed, and only the number of unique vulnerabilities may be used for compared.

The use of goal-oriented analysis is a unique feature of STRAP. This technique strikes a balance between the thoroughness and rigor of the more formal frameworks such as *i** and the Patrick & Kenny frameworks, while requiring relatively little training or effort to conduct. The primary reason for basing the analysis on such a structure is to avoid the danger of design-fixation. Whether the reasoning behind these design decisions is sound will hopefully become clear in the user-experiments discussed in chapter six.

5.2 Privacy-Aware Browsing: A STRAP Case Study

The preceding sections offered a crash-course in STRAP and its use in privacy aware design. In the next two subsections I present a detailed case-study illustrating the use of STRAP, meant to serve both as an illustrative example of how to apply STRAP, and to demonstrate that STRAP can be used to analyze large and technically complex domains. The next section gives an account of how this analysis was used to design and implement a privacy awareness and management system named iWatch.

This case-study looks at the Browsing domain, meaning the collection of web-browsers, web-sites, web-servers, and relevant parts of the communication infrastructure between these. In this section the domain is identified as Browsing as opposed to "browsing", which refers to the act of browsing. Certain browsers may not be prone to all the vulnerabilities discovered in this analysis, or they may contain unique features which introduce new vulnerabilities. Despite this abstraction, in part necessary because of the many different alternative products out there, this analysis is constrained to the realities of existing technologies and standards.

Because this is a very familiar domain and set of tasks for everyone, no detailed scenarios or sets of requirements will be given in advanced. The reader is assumed to be familiar with the workings of a modern web-browser (as of 2005), though not necessarily with how they work under the hood, the intricacies of communication protocols, or the causes for privacy vulnerabilities. Some, though not all of these, will be discussed in the text.

Though most of the syntax to be used in the goal decomposition and resulting goal-tree has already been discussed in section 5.1.1, it is important to identify the agents or actors in the system. In the Browsing domain the agents are the user, the browser, the web-server (and by extension the web-site), and the underlying network infrastructure, including DNS servers, TCP/IP protocols and the internet itself. These last two could be broken down further, but such an analysis would not have been a good time investment, as altering their behaviors are going to be out of the scope of this analysis.

We also need to introduce a new agent in the design and refinement-phase, iWatch. Recursive calls, calls to goals which are already decomposed elsewhere are marked in gray. The goal decomposition which is deferred is marked with a dashed

outline, and the goals which are used recursively elsewhere are given a bolder outline in the graph to make it easier to identify them.

Each of the agents in this domain gets assigned its own color in the graph. This allows us to tag the leave-nodes in the goal-tree according to which agent is responsible for ensuring that goal is carried out (see figure 18 for color key). In order to keep track of which vulnerabilities are addressed and which vulnerabilities are introduced in the different phases of this analysis, the call-outs describing these are marked with different types of hash-marks.

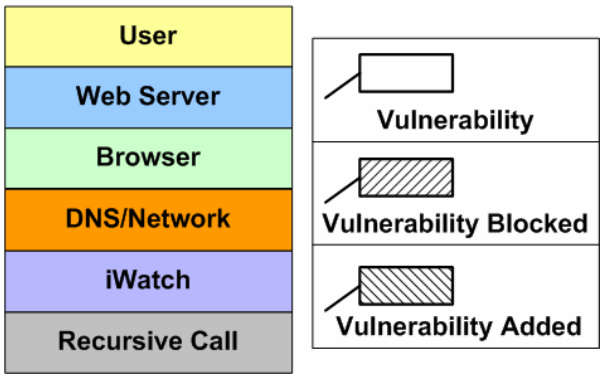


Figure 18: Color Key for Privacy-Aware Browsing Goal-Tree

Because of the size of the graph, it is divided into five overlapping figures, one for the top-level organization and one each for the four second-level goals (the goal "*Resource Viewed*" was not decomposed further. Each of these goals will be decomposed and discussed in detail in its own subsection. Figure 24, at the end of this section, gives a graphical overview of the goal-trees for both the decomposed Browsing domain, and the proposed design for a privacy-aware browsing domain.

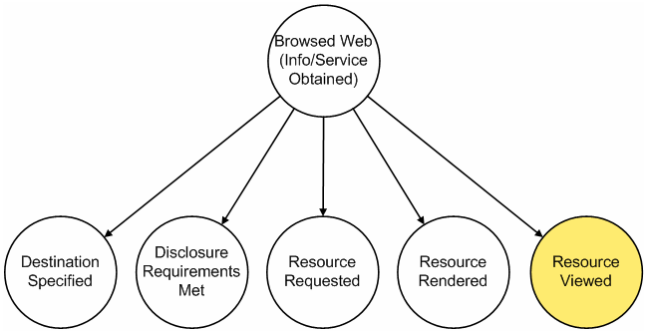


Figure 19: Top-Level Decomposition of Browsing Goal

The analysis starts by looking at the high level goal of "*Browsing*" and defines it as: wanting to gain some information or conduct some transaction with a web-site, which are the most common tasks users engage in online. If we break this down and think about the tasks or steps needed to accomplish this high-level goal we see that in order to obtain information or a service from a web-site a user needs to somehow specify the site they want to go to, disclose any information necessary for the transaction to take place, submit the request, wait for the result to load, and finally view and interpret the result. The top level goal of Browsing is therefore broken down into five sub-goals (Figure 19).

The last of which sub-goal, "*Resource Viewed*" is assigned to the user and not decomposed further because it represents an internal cognitive process of the agent. This does not mean that such a goal would never be decomposed or of interest to an analyst. In this case however we are not interested in analyzing how users interpret or understand the content of web-pages, making such a decomposition irrelevant in this case.

5.2.1 Destination Specified

In most situations, users have a great number of different ways of indicating or selecting what web-site or resource they wish to gain access to. The first step in this process however is always constant, and that is for the user to decide where to go. The user can then typically chose to select a bookmark (or some derivative of the bookmark),

type in a URL, select a recently visited site from the history list, or follow a link on a page. Note that in each case, the user will only do one of these, therefore these goals are marked with an OR arch.

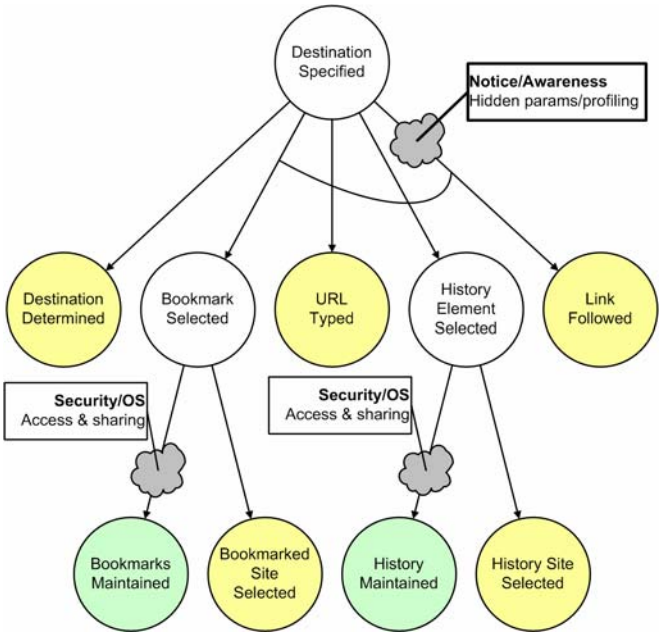


Figure 20: Destination Specified

Two of these sub-goals, "*Bookmark selected*" and "*History element selected*" require the system to keep a list of addresses that the user either considers important or has recently explored. This information, kept on the local machine, must be protected from other users by either the operating system or the browser (keep identities separate and protected when dealing with multiple local users). When following links it is also possible for the sites to pass hidden information about the user. Most browsers do this automatically through the referrer tag (telling the server what page the user was on when they followed the link). This need not always be a problem, but may warrant the users' attention.

Most of these goals can be broken down further (maintaining a bookmark file requires the browser to interact with the operating system to create, read, and write to a

file, requires the browser to draw up this list on demand etc.). This functionality however is simple enough that we can consider them as atomic, and say that other than the vulnerabilities associated with keeping this information somewhere on the local system, no other vulnerabilities exist. This is a judgment call of course, and different analysts will choose differently at this point.

5.2.2 Disclosure Requirements Met

In order to obtain a service or information it is often necessary for users to disclose some information about themselves, their interests, or the activity they are engaged in. Examples of this are logging into an online banking site, submitting a set of key-words to a search-engine, or submitting payment details to an e-commerce website. This goal can be decomposed into three sub-goals, a process by which the users decides what information to disclose, the act of disclosing the information itself, and the mechanical preparation of that information for transmission, in this case by appending it to a HTTP header. In an effort to save space, the final goal is presented between the two others in figure 21.

The goal of minimizing exposure is one which most users hold, even if they fail to act on it. In an ideal world this goal would compel users to perform a cost-benefit analysis of their actions before disclosing any information. In order to perform a cost-benefit analysis, users need to determine what the risks of disclosure are, understand the implications, determine potential benefits, and whether the tradeoff is in their interest. Here we find a number of vulnerabilities associated with the fact that most users have a limited understanding of the short-term or long-term risks associated with information

disclosures. This is compounded by the fact that users don't have insight into how their information will actually be used.

Users have two common sources for information about a site's practices, and therefore what risks they are exposing themselves to; a site or company's reputation, and the privacy policy posted on the site. Reputation tends to be by far the more influential factor in this equation as discussed in chapter 4.3.

Privacy policies, when present, have a number of important short-comings, as discussed extensively in chapter 4.2. The more relevant of these shortcomings for the purposes of this analysis includes the common practice of assuming consent from site visitors, regardless of whether they have read the policy or not. Policies are seldom read by users because they are often perceived as being too much work to parse, do not contain the information users are most interested in learning, and often contain inconsistencies (Earp et al., 2005).

Privacy policies are also typically disconnected from the implementation of the system. This means that there can be a gap between what the policy states and what the system does. With no independent verification of privacy practices, there is a relatively high risk of policies being incorrect or incomplete.

Once a user decides to disclose information, there are a number of mechanisms for doing so. The most straightforward of these is to type the information, but most browsers offer to recall and auto-complete information for the user. These auto-complete functions require storing information about the user on the local machine, information which must be protected from other local users.

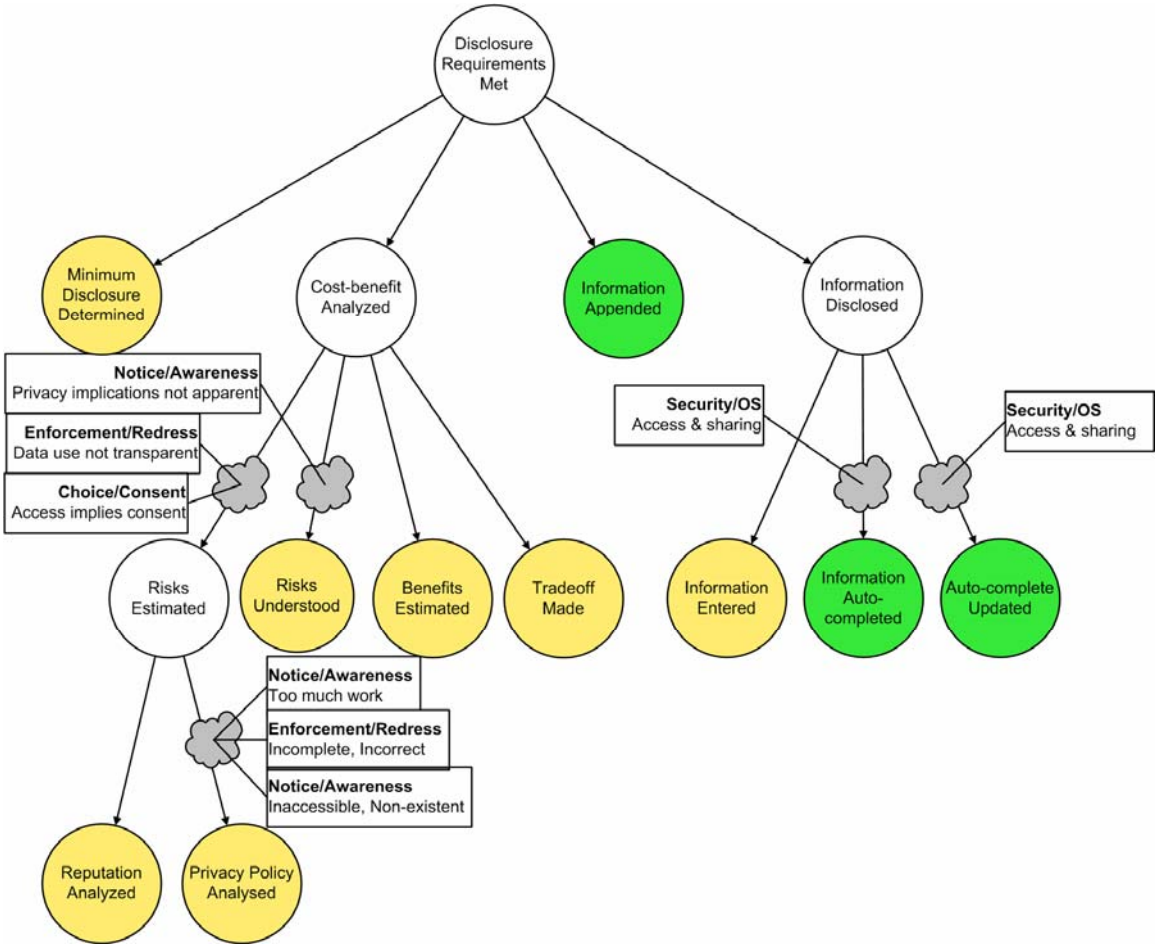


Figure 21: Disclosure Requirements Met

5.2.3 Resource Requested

Once a user has decided and specified what page to go to, and any disclosure requirements met, the page must be requested by the browser. In order to make this process efficient over low-speed connections, modern web-browsers employ a variety of caching strategies. This caching naturally leaves residue on the computer which must be protected from other local users. These cached files are less visible to users than for instance bookmarks. It is therefore likely that users are unaware of the information trail they are leaving behind. If a page is not found in cache it must be requested. In order to request the resource the HTTP request must be formulated, the server must be identified, and the request sent.

In addition to any information the user has supplied as part of the "*Disclosure Requirements Met"* goal, the request includes information which most users are unaware of, and lack control over. As part of the HTTP header, the browser will send any cookies associated with the site. The browser will also volunteer other information such as the referrer (the URL from which a user followed a link), operating system and details about installed components on the users computer. Users are largely unaware of this, and lack effective controls for changing this behavior. Much maligned, cookies in this context pose a danger to the user because they may reveal a users' past behavior to other local users. Cookies also pose problems because users are often unaware of them, or what information they contain.

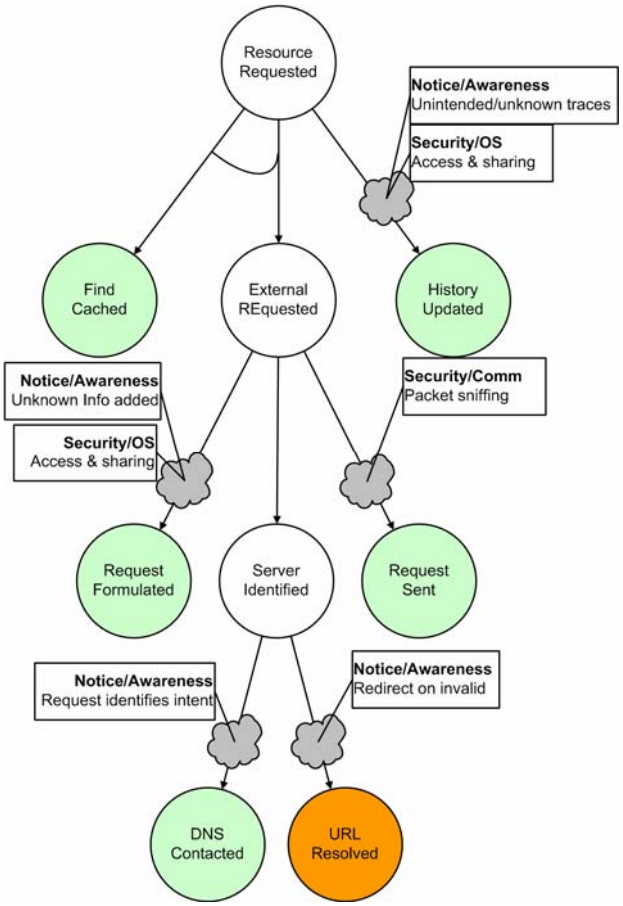


Figure 22: Resource Requested

In order to send a request, the browser must first figure out where to send it to. This can involve requesting DNS services from the ISP or other servers. While necessary, this is just one of the ways an outside entity can track a users' interests and behavior. In some cases, mistyping an address or following a bad link will redirect the user to a search page such as the default MSN search page for all users of the Microsoft Internet Explorer browser. This behavior, which users cannot easily alter, helps spread information about user habits to unrelated third parties.

Once the request has been formulated and the recipient identified the information must be transmitted. The HTTP protocol is not encrypted, and the internet is designed in such a way that messages are passed through a, from the users' perspective, arbitrary set of servers before it reaches its intended destination. If sensitive information is to be transmitted, adequate encryption or other security mechanisms must be in place. Furthermore, the user needs to be able to determine if such mechanisms are in place and evaluate what risks they are exposed to.

Finally, most browsers will make a note of the fact that the user has requested a given page, adding this to the browser's history file. Again, this information must be protected from other local users, and the user should be made aware of this fact and be given adequate controls over this behavior.

5.2.4 Resource Rendered

Once a resource has been requested, the user must wait for the result to be shown in the browser. In order for the result to be shown, the page must either be loaded from cache or received from an external server, and the browser must render the results on-screen. Cached pages, while speeding up the process of browsing, leave traces of the

user's past activities, often without their knowledge. This information needs to be protected from other local users.

When a resource is not found in the cache, or the cached copy has expired, the page must be requested and received from the server. In order for this to happen the web-server in question must receive the request, process it, and send the result to the client. When a server receives a request, this request is most commonly logged, a practice which many users are unaware of, and which must be protected.

Most modern web-servers process requests through a set of steps; identifying the recipient (IP, port), extracting any request parameters (cookies, forms, etc), identifying and processing the resources to serve, and formulating the response to send. Assuming the company is basically honest and abides by its privacy-policy, and that this policy contains a complete and accurate disclosure of their privacy practices, there are three vulnerabilities to end-user privacy. First, any information transmitted to the server as part of a transaction or request must be processed and stored securely. Second, users need to be aware of how information accumulates over time, and how small, seemingly innocent and unimportant pieces of information can add up to a detailed profile. Finally, users need to be protected or given control from hidden redirects. This is a practice which forces resources (web pages) on users without their knowledge or consent, thereby invalidating their risk estimates. Once a response has been generated the response must be sent to the user. This communication may need to be protected in order to prevent unauthorized interception and tracking by third parties.

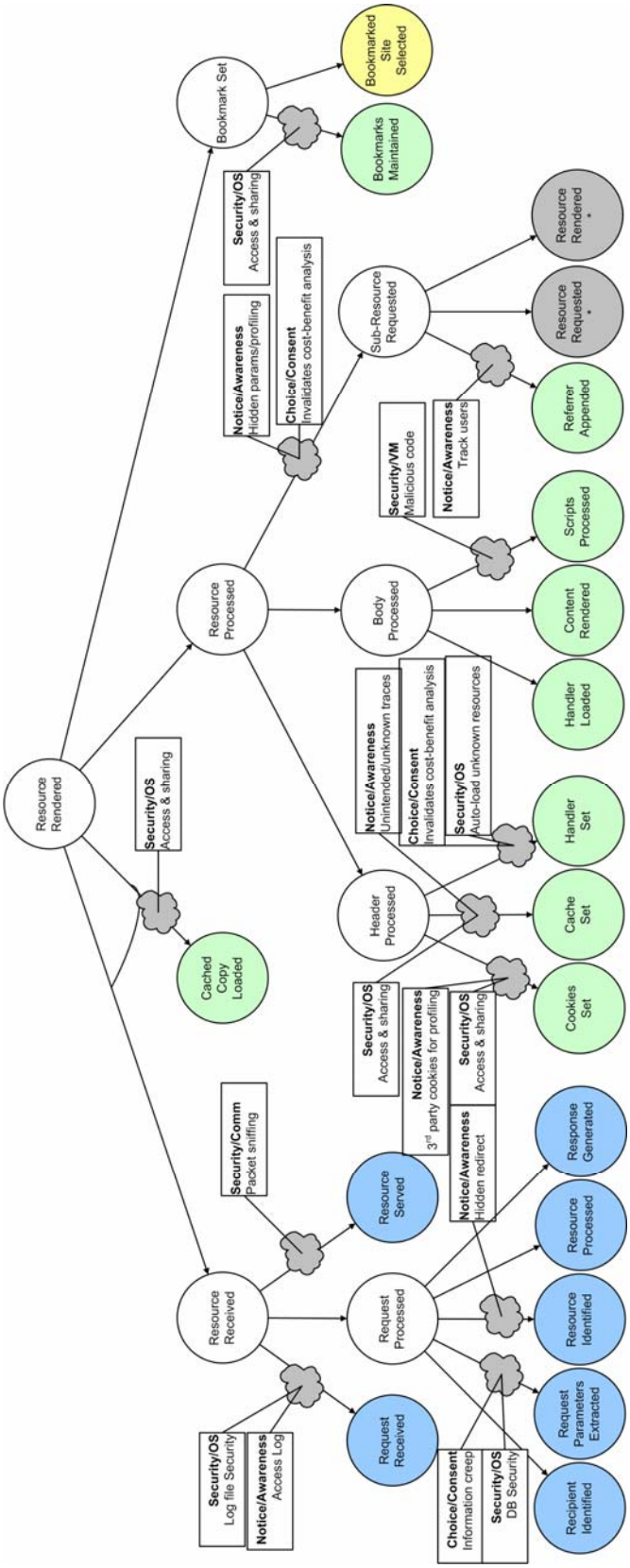


Figure 23: Resource Rendered

Once a page has been fetched from cache or received from a server it must be processed by the browser. This means the browser must process the HTTP headers, the body containing the actual HTML to be rendered, or in some cases generated, and request any additional resources which may be requested.

When processing the header, the browser needs to set cookies, determine if the page should be cached, and load any plug-ins or auxiliary programs needed to process the body. Handling cookies has already been discussed in section 5.2.3, as has the caching of pages in 5.2.4. Loading auxiliary programs can invalidate the cost-benefit analysis the user performed, as different media may expose them to different risks. With embedded content such as Macromedia Flash or ActiveX applications, users often do not know ahead of time what they will be loading. Loading unknown media types may trigger the browser into automatically searching for or triggering the download of handler programs, or expose the user to new security and privacy risks.

When the browser gets to processing the body of the page any scripts associated with the page are executed, media and other special content is passed to the appropriate auxiliary program, and the content itself is rendered. The only new vulnerability in this process is a security problem in that scripts may be malicious in nature, compromising the users' security or data.

Any page may also include or call on additional resources from the same or other servers, which may expose the user to unknown risks, or serve to provide information to third parties. Examples of this are inline images, script files or web-bugs. Users have little if any knowledge or control over whether such resources are loaded, or even an opportunity to include this knowledge in their initial risk analysis. Furthermore, it is

possible for the service providers to pass information about the user as parameters to the request, thereby facilitating data-sharing and tracking between them.

The final step in the process is to allow users to bookmark the page, the mirror image of the process employed in selecting where to go to, with the same vulnerability, the fact that this bookmark file must be protected from other local users.

5.2.5 Analysis

The preceding sections present an in-depth analysis of the goals and vulnerabilities associated with the task of browsing, as seen from the users' perspective. Of course, a large number of other functions are needed to ensure that these systems actually work (such as a working internet infrastructure, operating system support, a number of and most server functions). Most of the vulnerabilities discovered have at one time or another been documented by others, and a collection of these have been addressed by different privacy-protection or management systems.

In a sense, the results of this vulnerability analysis are not revolutionary; very little was discovered which was not already known. What is unique to this analysis is that it provides a systematic listing and organization of all known privacy vulnerabilities associated with browsing. Such a listing has been missing, and as a consequence many researchers and developers end up talking past each other when they discuss different strategies for online privacy management. By presenting a clear and unified picture of how the different vulnerabilities interact and fit into the overall system, it is possible to see how these different systems and strategies proposed interact, overlap, and partition the problem space.

Table 11: List of Privacy Vulnerabilities Discovered in Browsing

Count indicates how many occurrences of the same problem, or same class of problems

Description	Count
Security/OS: Access & Sharing	10
Security/Communication: Packet sniffing	2
Security/OS: Log file security	1
Security/OS: Auto-load resource handler	1
Security/VM: Malicious code	1
Security/OS: Database security	1
Notice/Awareness: Hidden parameters	4
Notice/Awareness: Unintended/unknown traces	3
Notice/Awareness: Hidden redirect	2
Notice/Awareness: Implications not apparent	1
Notice/Awareness: Notice too complicated	1
Notice/Awareness: Notice inaccessible/non-existent	1
Notice/Awareness: Access implies intent	1
Notice/Awareness: Access log kept	1
Enforcement/Redress: Data use not transparent	1
Enforcement/Redress: Incomplete/incorrect notice	1
Choice/Consent: Hidden-resource invalidates analysis	2
Choice/Consent: Notice access implies consent	1
Choice/Consent: Information creep	1

This analysis uncovered a total of 19 unique vulnerabilities, as listed in table 11, grouped by FIPs category. Looking at the diagrams, one will see that a total of 36 vulnerabilities were identified. This number was inflated because many of these vulnerabilities were duplicated. These vulnerabilities are grouped according to their FIPs categorization. Many of these vulnerabilities can be placed in multiple FIPs categories, and this classification should only be used to suggest mitigation and elimination strategies.

Though the statistics presented here regarding the number of vulnerabilities, or unique vulnerabilities are informative, one should not read too much into these numbers. The raw number of vulnerabilities, as opposed to the number of unique vulnerabilities, can be influenced by the way the analyst breaks down the problem. Decisions such as whether a local access and sharing problem gets counted when a file is written, read, or both can lead to one or two vulnerabilities reflecting fundamentally the same problem. What is an analyst includes goals to create the file if the file does not already exist? This could mean a third vulnerability, etc.

Counting unique vulnerabilities is somewhat more informative, though the analyst must remain objective when grouping vulnerabilities. Is it the same vulnerability when we are talking about the access and sharing of cached IP addresses as when we are dealing with bookmarks or cached content? The underlying mechanism causing the problem is the same, but we are dealing with very different types of data, collected and used for very different purposes. These are decisions which the analysis team must make depending on their needs and existing practices. In the preceding example we decided to lump vulnerabilities together when their underlying causes were the same, regardless of differences in the information collected.

5.2.5.1 Security Vulnerabilities

A total of 16 (44.4%) of vulnerabilities can be classified as security vulnerabilities, though only 6 of these were unique, accounting for 31.6% of all unique vulnerabilities. These are very high numbers considering that the heuristics employed are primarily aimed at identifying awareness and control issues. Though security and privacy are two separate requirements, it is easy to see how much of a pre-requisite security is to privacy.

The problem associated with the storing of cookies (needing to keep them safe from other local users) is listed twice; once in “Resource Requested” when a cookie can be read, and once under “Header Processed” when cookies can be set on the users machine. This vulnerability (the need to protect information and resources from local users) is something we see repeated in a number of different goals, such as those dealing with bookmarks, history files, caching of resource, and auto-complete information. In all, a total of 10 (27.8%) vulnerabilities were of this class (two for most types of resource, once when read, and once when written).

The vulnerabilities associated with the storage of different types of information, for different purposes, and by different system modules, can often be addressed through the same mechanisms. Furthermore, if one assumes the browser is running on a secure multi-user platform, this vulnerability disappears as the OS guarantees that no local users can access the data. If such an assumption cannot be made, the system will need to address these vulnerabilities through mechanisms such as encryption, or eliminate these problems altogether by not caching or storing information locally.

There are another six security vulnerabilities, two dealing with the protection of (not only encryption, but also the establishment and enforcement of good data access

policies) information on the server (logfiles and databases in “Resource Rendered”), two dealing with the secure transmission of data (“Request Sent” under “Resource Requested”, and “Resource Served” under “Resource Rendered”). These four vulnerabilities only pose problems when the system is not using available encryption mechanisms and modern operating systems. The final two security vulnerabilities deal with the potential for malicious code, an ongoing problem for all browsers.

5.2.5.2 Awareness Vulnerabilities

Another major portion of the vulnerabilities discovered (14, 38.9%) can be classified as vulnerabilities arising from a lack of notice to users, limiting their awareness of risks. Of the 14 vulnerabilities, eight were unique, accounting for 42.1% of all unique vulnerabilities. These vulnerabilities need not always present a problem to the user, but rather present opportunities for users to make erroneous judgments or miscalculations. These vulnerabilities in particular pose interesting HCI challenges, how to provide adequate warning and information without over-burdening or being overly distracting.

The first set of these awareness vulnerabilities are closely related to the security vulnerabilities discussed previously. In five instances, the browser may store or retrieve information on the users’ computer without the users’ knowledge or intervention. While done with the users’ best interests in mind, these strategies can blindside the user to the risks they may actually be exposed to. The rest of these awareness vulnerabilities are hidden or obscure practices which may affect the users privacy, and which the user has no reasonable way of discovering or monitoring. These are largely centered on the possible use of hidden parameters in links and between sites, details about the workings of TCP/IP and HTTP, and uncertainties surrounding server-side data practices.

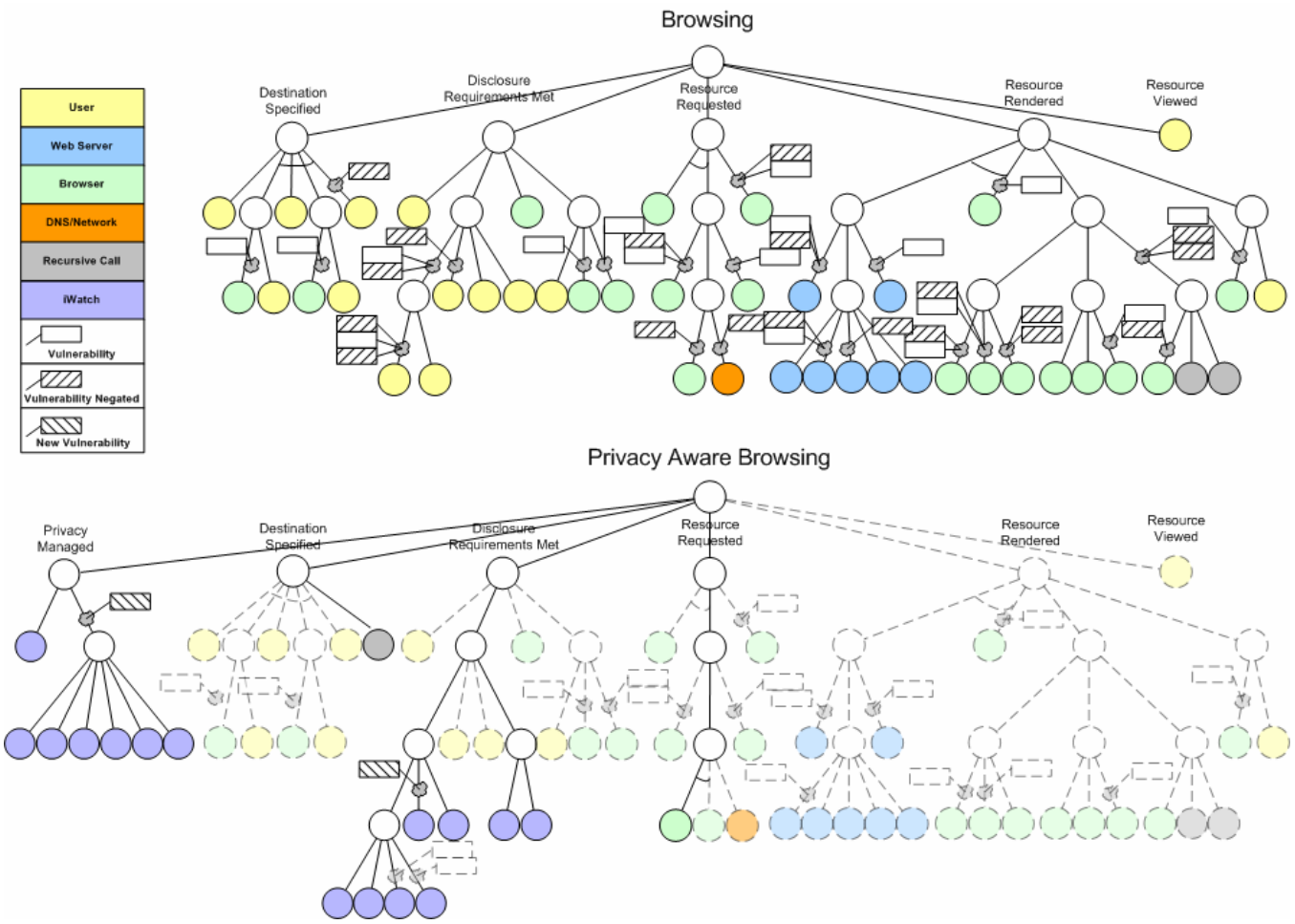


Figure 24: From Browsing to Privacy-Aware Browsing
Faded goals in Privacy Aware Browsing are unaltered from Browsing. Note new recursive goal in Destination Requested. Vulnerabilities marked as being added, removed, or unaffected by their coloring

5.2.5.3 Enforcement Vulnerabilities

Two vulnerabilities (5.6% of all vulnerabilities, or 10.5% of unique vulnerabilities) in "Disclosure Requirements Met" could be classified as enforcement vulnerabilities. These vulnerabilities stem from the risk-benefit analysis and the absolute lack of transparency in organizations' data practices. In order for users to evaluate whether a site is fulfilling its commitments, or seek redress when mistakes or violations have occurred, they need access to basic information about their records and how their information is being used. Some legislative measures have been introduced to change this situation (such as the EU Data Directive (EU, 1995)), but it is not yet a prevalent practice.

5.2.5.4 Choice/Consent Vulnerabilities

The final four vulnerabilities were classified as one of three unique Choice/Consent vulnerabilities (11.1% of all vulnerabilities, 15.8% of all unique vulnerabilities), and refer to situations where the browsers default or automatic behavior pre-empts end-users decisions.

5.3 From Privacy-Aware Browsing to iWatch

Once a set of vulnerabilities have been identified, different design solutions should be explored. As described earlier, elimination strategies primarily revolve around the removal or substantial alteration of goals to avoid these problems. Mitigation strategies revolve around augmenting the goal-tree to include goals and functions which will make vulnerabilities less likely to occur or ensure that the damage is minimized.

This section is divided into three sub-sections, a discussion of a design solution, a goal-refinement dubbed Privacy-Aware Browsing, an analysis of Privacy-Aware browsing and how the these changes affect the rest of the system and assumptions made, and finally a discussion of how Privacy-Aware Browsing was transformed into a working prototype called iWatch.

5.3.1 Defining Privacy-Aware Browsing

Given that we are dealing with an existing set of systems which have a very large user-base, most negation strategies are unrealistic. We could come up with a design calling for substantial changes to the way TCP/IP works, or modify the way in which browsers interact with servers, but this is not realistic or valuable, even as a theoretical exercise. In this section I will therefore primarily pursue mitigation strategies and describe a set of design changes which will address more than half of the known vulnerabilities.

Though the proposed design solution, dubbed “Privacy-Aware Browsing” (PAB) will be discussed in-depth in the next section, an overview of the changes from Browsing to PAB is shown in Figure 24 for the readers’ convenience. Note that relatively few parts of the goal-tree are affected, yet the effects ripple through the whole tree.

In choosing which vulnerabilities to address one has to determine where to best invest ones efforts, and set limits according to the available time, manpower, and expertise. We must also identify what our operating assumptions are going to be in terms of browsers, operating systems and other underlying technologies, thereby determining which of these vulnerabilities affect us, and which are addressed by other technologies. For the purposes of this case-study we will focus on the Microsoft Internet Explorer 6.0

browser running on Windows XP, both running with service pack 1, and connecting to a normal TCP/IP network. This choice of operating system and browser was made because this was by far the predominant configuration at the time. We do not make any assumptions regarding the server environments, and indeed limit ourselves to addressing problems on the client-side.

Without getting into a lengthy debate about the level of local file-system security offered by Microsoft Windows XP, we acknowledge that this vulnerability is not fully addressed by the operating system. Microsoft has taken steps to harden its products through periodic patches, and we can therefore assume two things; that this product has matured to the point where the user is at least offered some protection, and that we probably cannot do a better job with the resources, manpower, and experience at hand. Instead we turn our attention to the second largest category of problems, notice/awareness, problems we are much better equipped to address.

Another reason for ignoring purely security-related vulnerabilities in this design solution is that (i) these problems are receiving or have received significant attention by people with more of a background in security and security research. STRAP is not aimed at addressing the concerns of these practitioners; we defer these problems to others. (ii) The remaining purely privacy related problems can typically not be addressed through security measures and to a larger extent require the involvement of the end-user, making for more interesting problems for us to deal with.

Looking at table 11 we see that 11 of the 19 (57.9%) unique vulnerabilities identified in the analysis can be classified as awareness or control problems (Notice/Awareness or Choice/Consent). This is encouragingly consistent with the data found in the studies presented in chapter four. Most of these vulnerabilities stem from an

inadequate level of support in evaluating the risks associated with visiting a site or with disclosing information to a site. Part of the problem is a lack of information about what sites are doing, and partly a lack of awareness of how and what data and patterns of behavior teaches others about us. The Privacy Aware Browser is our design proposal for how to address these problems.

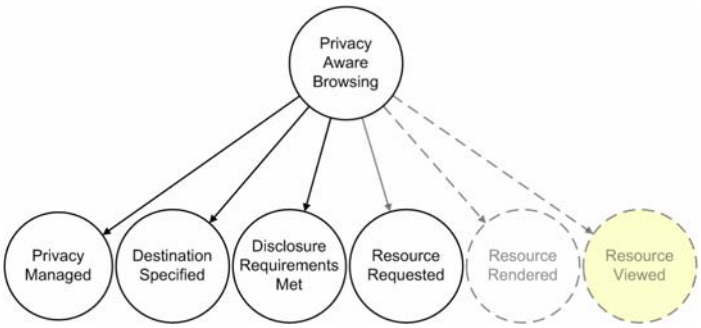


Figure 25: Top-level PAB

Figure 25 shows the top level decomposition of PAB. A new second level goal has been added to Browsing (*Privacy Managed*), and *Destination Specified*, *Disclosure Requirements Met* and *Resource Requested* have undergone changes. *Resource Rendered* and *Resource Viewed* remain unchanged, in terms of goals and functionality, though not in terms of vulnerabilities. Note that all goals which have not been affected by this design change are faded and their outlines dashed to better focus the readers' attention on the relevant parts of the goal-tree.

To address these vulnerabilities it is essential to provide users with adequate decision-support tools. A large part of this consists of keeping track of what the user has done in the past so that adequate context may be given, and more advanced risk assessments may be done. The addition of the *Privacy Managed* goal (Figure 26), allows the system to provide the user with a persistent memory from which to give assessments and warnings about how past actions affect current or future transactions, what risks they

have taken, and the facility to track leaks and information misuse. This of course requires storing some information on the users' machine, which in turn introduces a new security vulnerability. This is an acceptable tradeoff, given the number of vulnerabilities which can be mitigated with this new functionality.

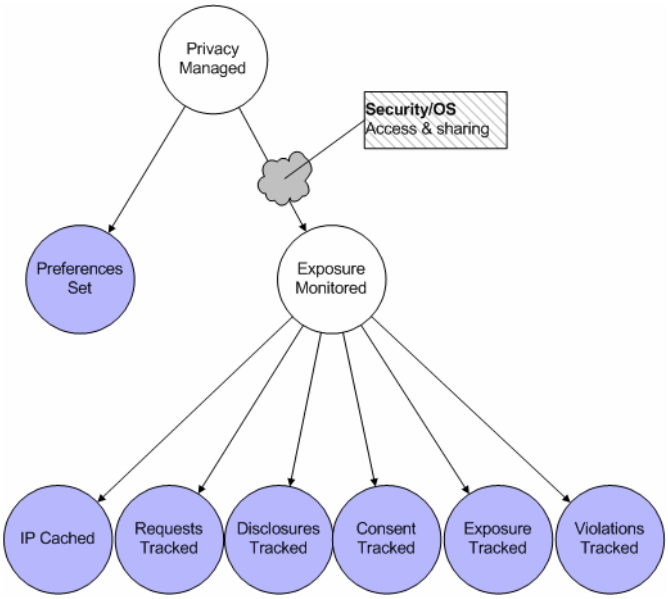


Figure 26: Privacy Managed - PAB

We also include facilities for setting preferences as well as caching IP addresses. This last goal is already performed by the operating system, and prevents a DNS server from tracking every site the user goes to. This goal has its mirror image in *Resource Rendered*, which is modified to look in the cache before contacting the DNS (figure 27.)

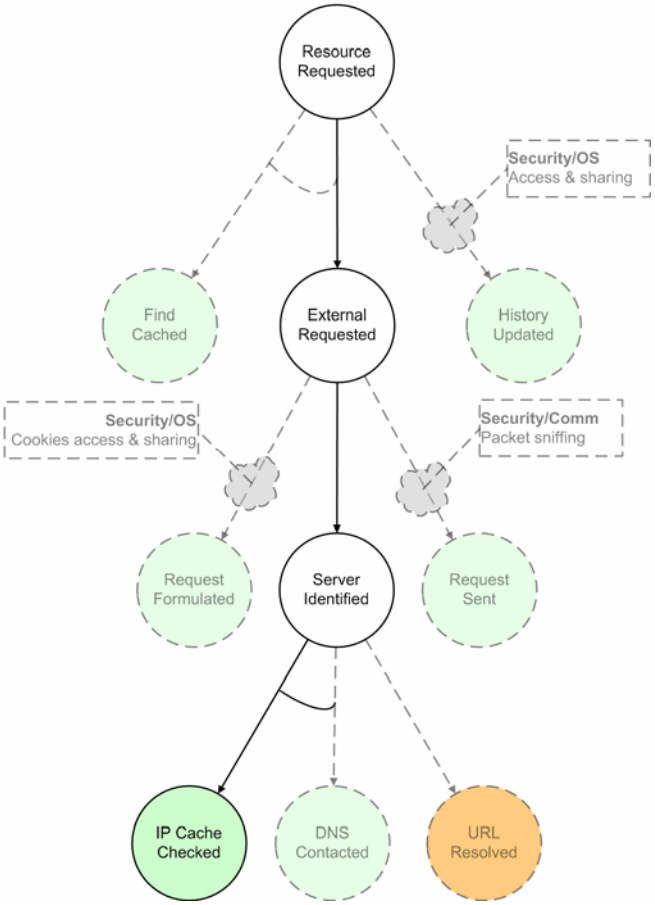


Figure 27: Resource Requested - PAB

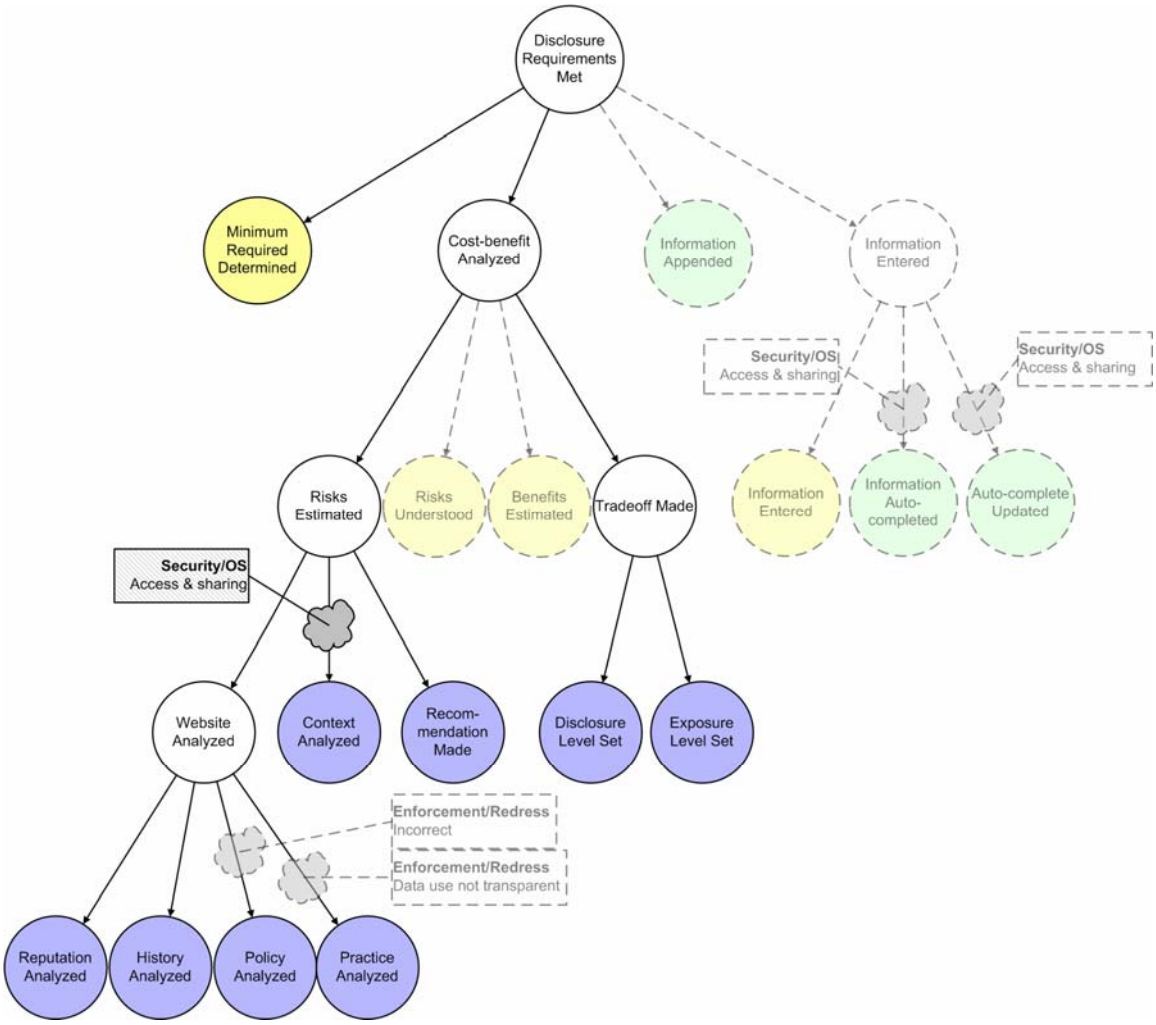


Figure 28: Disclosure Requirements Met - PAB

The second major change in the design is a revamp of the decision-support in *Disclosure Requirements Met* (figure 28). Previously, risks were estimated by the user without any support, and with only the sites' policy and reputation as guides. In PAB, the browser supports the user by performing an analysis of machine-readable policies, by looking at what practices are encoded in the page, and by keeping track of a sites' reputation among similar users and any past infractions of policies. With this task being off-loaded to the system, the user can rely on setting preferences or thresholds for things such as warnings or automatic filtering of pages or behaviors based on preferences set in the *Privacy Managed* goal. Two of the vulnerabilities previously associated with the goal *Risk Estimated* are now moved down to their more appropriate sub-goals, but have not been altered, and are therefore faded.

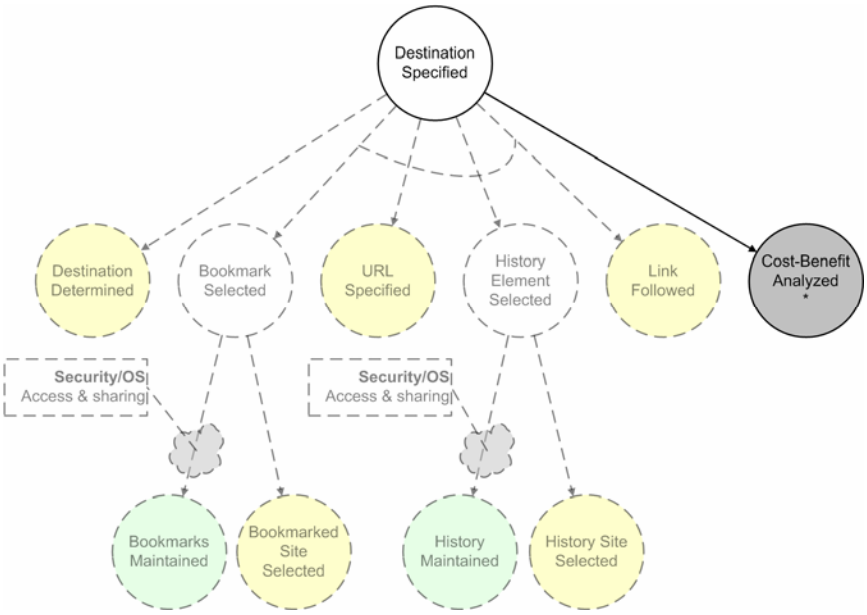


Figure 29: Destination Specified - PAB

In addition to performing this cost-benefit analysis when disclosing information, the user, or the system should really be performing this when choosing what sites to go to in the first place. While most sensitive information requires the users' direct input in as

part of meeting a *Disclosure Requirements Met* goal, a not insignificant amount of information is disclosed when a user first visits a site. Many exploits rely on misdirection, hoping the user does not notice a redirection from a legitimate or legitimate-sounding site to a fraudulent site, or the 'phishing' for users' email addresses by embedding remote images or hyper-links in emails in the hope that these will be opened, thus passing the relevant parameters to those wishing to exploit the user. In order to make users aware of this fact and help decide whether they really want to assume that risk, the system should provide basic cost-benefit support at this stage as well. Figure 29 shows the *Destination Specified* goal-tree, which is unaltered from Browsing, except for a call to Cost-benefit Analysis as defined in *Disclosure Requirements Met.*

The final goal-tree, *Resource Rendered* remains unaltered in terms of goals. Of course, this does not mean that this goal-tree is unaffected by the changes. Looking at figure 30 we see that a large number of vulnerabilities in *Resource Rendered* are addressed by the changes made elsewhere in the goal-tree. The vulnerabilities which are addressed in this goal-tree are vulnerabilities which invalidated the users' risk estimates, or which exposed the user to risks they had no way of knowing about. By offloading these tasks to the system, more realistic risk-estimates can be derived because the system can check for sub-resource or the hidden dependencies and sharing practices of a page.

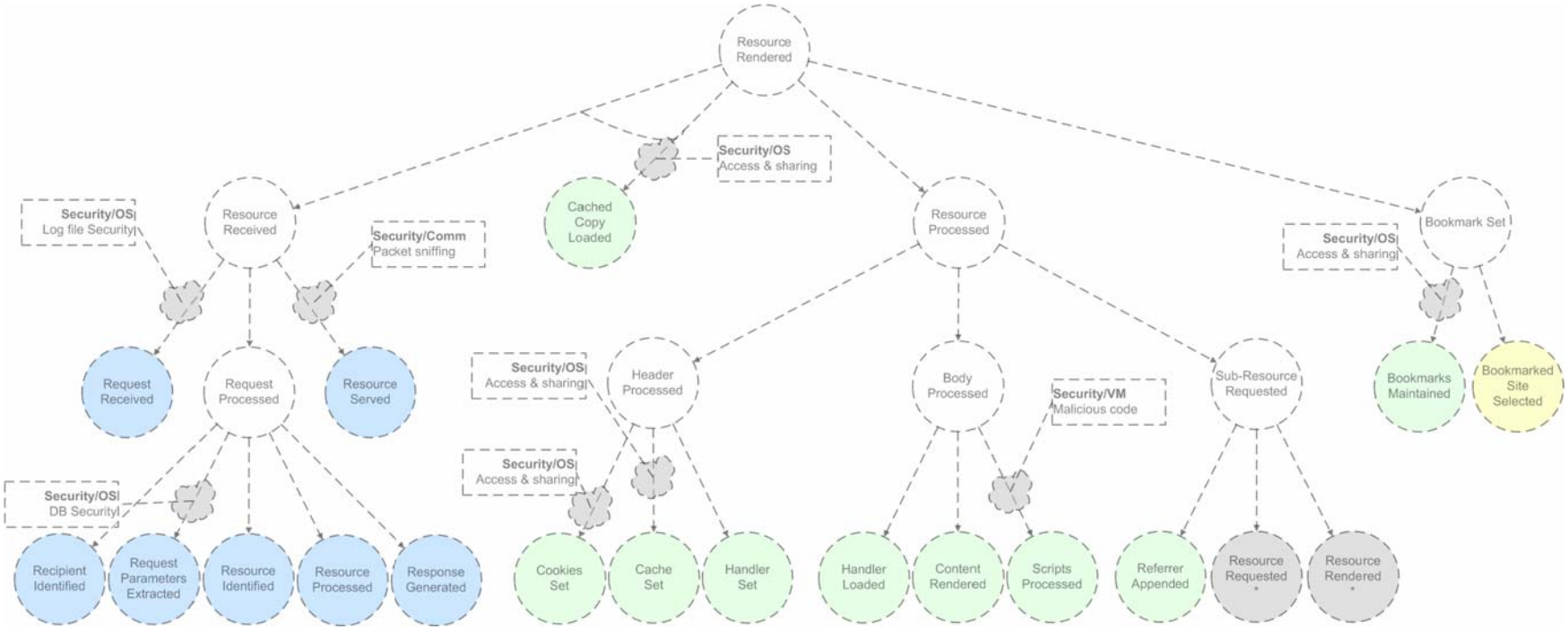


Figure 30: Resource Rendered - PAB

5.3.2 Analysis

In Privacy Aware Browsing I have only proposed three changes to the original Browsing goal-tree; a set of functions to help users track their information disclosure and the risks and practices they have encountered, a more rigorous and automatic risk estimation, and the caching of IP addresses to limit the involvement of third-parties in the process.

This definition of PAB does not address all the vulnerabilities found in Browsing, or affect all the goal-trees. Because no new goals of goal-redefinitions take place in the rest of the goal-structure does not mean that these are unaffected. To determine what effects any change has, an analyst should walk through the entire goal-structure, considering how each goal and vulnerability is affected. Only new goals, or redefinitions of old goals may affect how or whether vulnerabilities occur, and given that the analyst already has the structure, this is a relatively simple task to perform, and is the same one performed at the end of each design iteration.

It is interesting to note that all these vulnerabilities have been mitigated, and none have actually been eliminated, and artifact of the little control we have over the architecture. The strategy employed has been to collect information, to provide decision support rather than pursue strategies which would have made it impossible for users to have their information stolen or misused. This is partly an artifact of the fact that the infrastructure is so fixed that it is difficult to make fundamental changes to how the system as a whole functions. All that we can really do is try to support the users in making informed decisions. Ultimately, this is not something the system can do for the user.

We address (by mitigation) 19 of the original 36 vulnerabilities (52.8%), while adding 2 new vulnerabilities, leaving a total of 19 vulnerabilities in Privacy-Aware Browsing. Exactly how efficient these mitigation strategies will be is debatable, and should be examined further. These vulnerabilities cannot be discarded and should be part of the systems documentation. Overall, 12 of the 19 unique vulnerabilities (53.2%) were addressed in this design refinement. The two vulnerabilities added are related to the OS security vulnerability dealing with the local storage and potential compromise of data. By storing more data locally we increase the overall risk associated with this vulnerability. Table 12 lists the remaining vulnerabilities.

Table 12: List of Privacy Vulnerabilities Remaining in Privacy-Aware Browsing
Count indicates how many occurrences of the same problem, or same class of problems

Description	Count
Security/OS: Access & Sharing	12
Security/Communication: Packet sniffing	2
Security/OS: Log file security	1
Security/VM: Malicious code	1
Security/OS: Database security	1
Enforcement/Redress: Data use not transparent	1
Enforcement/Redress: Incomplete/incorrect notice	1

As we see, all that remains unaddressed are the security related vulnerabilities, and the enforcement/redress vulnerabilities. I have already discussed the problems associated with addressing the security vulnerabilities. The enforcement/redress vulnerabilities are likewise very difficult to address without a fundamental shift in the way online businesses operate because they would require unprecedented access into the sites' information handling practices. To eliminate these vulnerabilities, the data subject would need some mechanism to audit the companies' data practices to check that they are being honest. Technically, this is not terribly difficult to accomplish because many databases are set up to log queries. All that would be needed would be to provide

mechanisms for anonymizing this data and making it available to users in a way they would understand. Such a simple technical solution can result in major challenges, both in terms of privacy and organizational and business practices.

5.3.3 iWatch as a Privacy-Aware Browser

The previous section presented a design solution which would address a large number of the vulnerabilities associated with browsing through different mitigation strategies. This design solution, dubbed Privacy-Aware Browsing, took into account certain limitations in terms of existing infrastructure. From this we look at the costs associated with the actual implementation of these goals, and the resources available, and how Privacy-Aware Browsing can be implemented as part of our privacy awareness and management system, named iWatch.

From a look at the requirements and the limitations we have set ourselves, it becomes obvious that as we can neither modify the underlying OS, nor the browser, iWatch must be implemented as a proxy. A proxy server acts as an intermediary between the two, modifying and interpreting the communication and requests that are sent beween the two. A proxy server can also intercept and modify the requests a browser sends out, and filter the information a server sends to a browser before this has a chance to execute those instructions. In addition to filtering communication, as all proxy servers, iWatch will need to present the user with appropriate feedback and control options which will let him or her stay aware of what any website is doing, and manage their privacy.

There are a number of stable and well-documented open-source proxy platforms on which we can build, further reducing the costs of implementation. This allows us to focus our time and energy on the features which make iWatch unique rather than the

underlying functions needed to interact with the browser, the Internet, or perform basic filtering. After careful examination we decide to base iWatch on the Privoxy open-source project[3], which in turn is based on Internet Junkbuster[4], both of which operate under the GNU General Public License (GPL)[5] license. This proxy allows the user to specify very flexible filtering custom filtering rules, using regular expressions, and has a very minimum user-interface which can easily be replaced. This allows us to easily transform Privoxy into the foundation for iWatch.

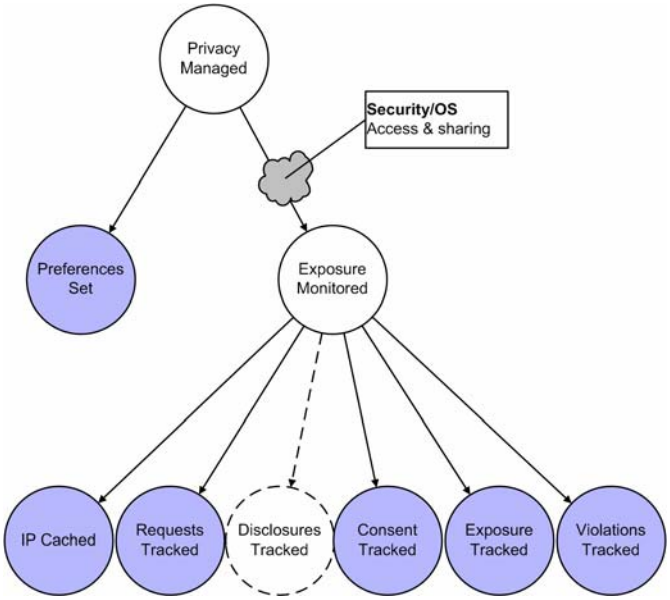


Figure 31: iWatch Privacy Management Implementation

Next we look at the goals associated with the new *Privacy Managed* goal-tree to see what functionality needs to be added. Most of the goals are relatively simple to implement. Because the proxy acts as an intermediary between the browser and OS, we have access to all the necsessary information. Doing the necessary analysis and tracking

---

[3] http://www.privoxy.org

[4] http:/internet.junkbuster.com

[5] http://www.gnu.org/copyleft/gpl.html

therefore relatively simple. The only goal which was deemed too costly to implement was to track information disclosures. While all communication is available to us through the Proxy, the ammount of processing needed to interpret and classify this information is far from trivial.

The goals described in the cost-benefit goal-tree that is part of the new *Disclosure Requirements Met* and *Destination Specified* are more complicated. These goals require both the interception and filtering of information.

The first goal we abandon is that of maintaining reputations for sites. This is a very expensive procedure because it requires a large initial investment in seeding the recommendation system. Other goals too expensive to implement given our resources were the context analysis and recommendation goals. Some context is analyzed but this is a difficult goal to fulfil properly. These would have required a significant investment in understanding the context and dangers associated with different actions. Figures 31 and 32 give an overview of which goals are implemented and which were abandoned because of cost. Abandoned goals are shown with a dashed outline.

iWatch is a combination web-crawler and a privacy proxy, with a management and awareness interface. An overview of the architecture developed is shown in figure 33. The crawler was a necessary addition to the architecture in order to provide advanced warning and the ability to perform historical analysis of sites' practices and policies. The crawler is meant to serve as a knowledge repository for the proxy, a source for information and early warning telling the crawler whether it should or should not contact the server in question. This negates the potentially tricky vulnerability of assumed consent and that just contacting a server potentially discloses a certain amount of information about the user and his or her habits and interests.

Figure 32: iWatch Disclosure Requirements Met Implementation

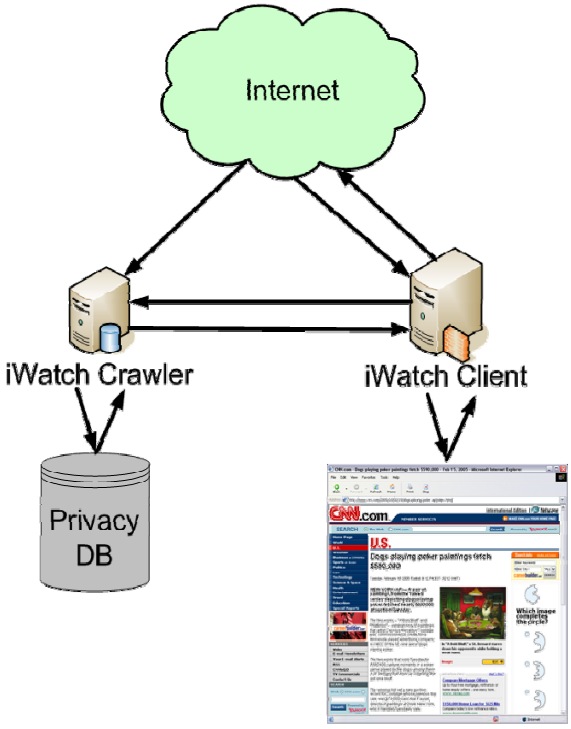


Figure 33: iWatch Architecture

The crawler runs on a central server and is accessible by all iWatch clients, which run on the users' machine. The web-crawler scans web-pages, analyzing their privacy practices (such as the use of cookies, web-bugs, P3P etc.) and provides a ready source for statistical information regarding any specific site or the state of the web, prevalent privacy practices, and emerging trends. An example of this type of analysis can be seen in figure 34, which shows how iWatch can detect hidden information sharing networks which exist between sites.

The crawler starts by looking at a list of sites in its database, seeded by the iWatch clients to ensure a relevant sample is analyzed. The crawler looks at these pages and scans for twenty-one common privacy practices which can affect the users' privacy (see Table 13). The crawler simply notes the existence of these practices, and how they evolve over time. The proxy looks for the same practices, and allows the user to specify which of these practices they want to be warned about, and which they want to block.

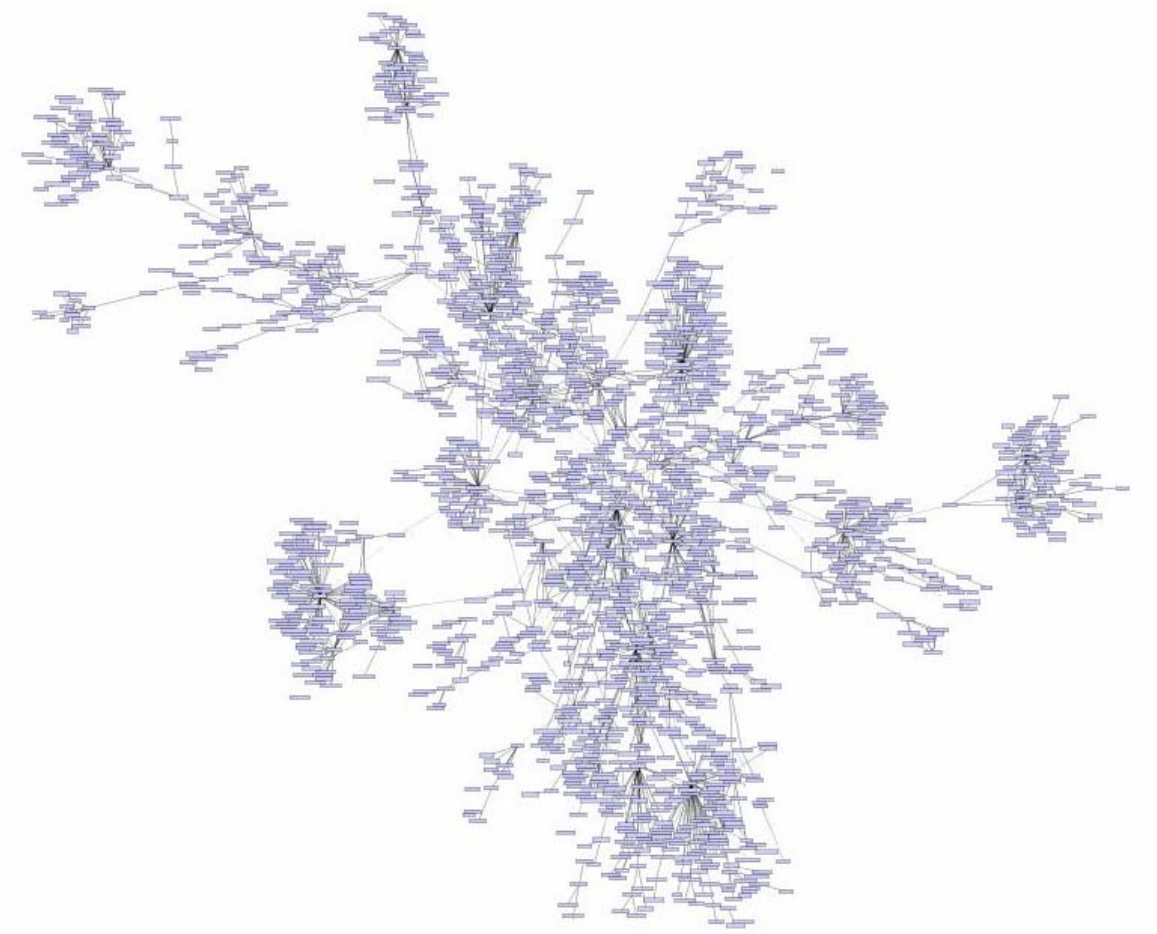


Figure 34: Hidden Information Networks Detected by iWatch

Each blue rectangle in this network is a website, each line an embedded images, third party cookies, or web-bug which allow these sites to share information or track users. 70% of the sites at the center of clusters are servers dedicated to advertising or market-research.

Table 13: List of iWatch Filters

Count indicates how many filters of this type. Filters grouped by function. Predatory practices as defined by Junkbusters and Privoxy

Filter	Count	Description
Cookies	3	Identifies the use of different types of cookies (session, normal and 3rd party)
Unsolicited popups	1	Identifies the use of unsolicited popup windows
Web-bugs	1	Identifies the use of third part resources potentially used to track users from site to site
Image reorder	1	Identifies image reordering and hiding, sometimes used to place web-bugs
Banners	2	Identify the use of different types of banners and adds, potentially used to track users from site to site
Full P3P	1	Identifies the use of full P3P privacy policies by site
P3P compact policy	1	Identifies the use of compact P3P privacy policies by site
Crude-parental	1	Crude parental filter looks for list of curse and pornographic words
Hidden forms	1	Looks for hidden forms sometimes used to pass along information without the users knowledge
Refresh tags	1	Identifies refresh tags sometimes used to redirect users or pass hidden information to websites
HTML annoyances	1	Identifies practice typically associated with predatory sites
Jumping windows	1	Identifies practice typically associated with predatory sites
IE-exploits	1	Identifies the use of known Internet Explorer exploits
Javascript annoyances	2	Identifies different types of known javascript exploits and practices typically associated with predatory sites
Shockwave/flash	1	Identifies the Macromedia Shockwave or Flash
Quicktime/kiosk mode	1	Identifies the use of quicktime and quicktime kiosk mode

The interface model employed in the proxy is a hybrid effect-driven and techno-centric model where users specify what thresholds they wish to set to specific technologies or effects. The proxy also provides two awareness UI's which integrate into Microsoft Internet Explorer browser. These UI's are meant to provide users with information about the privacy practices of the site they are visiting, according to the preferences the user has specified. These two UI's differ in terms of the amount of detail they offer users and the amount of screen-real-estate they require (See figure 35). This allows the user to roughly decide how much distraction they are willing to tolerate, and how much detailed information they are interested in, or find useful.

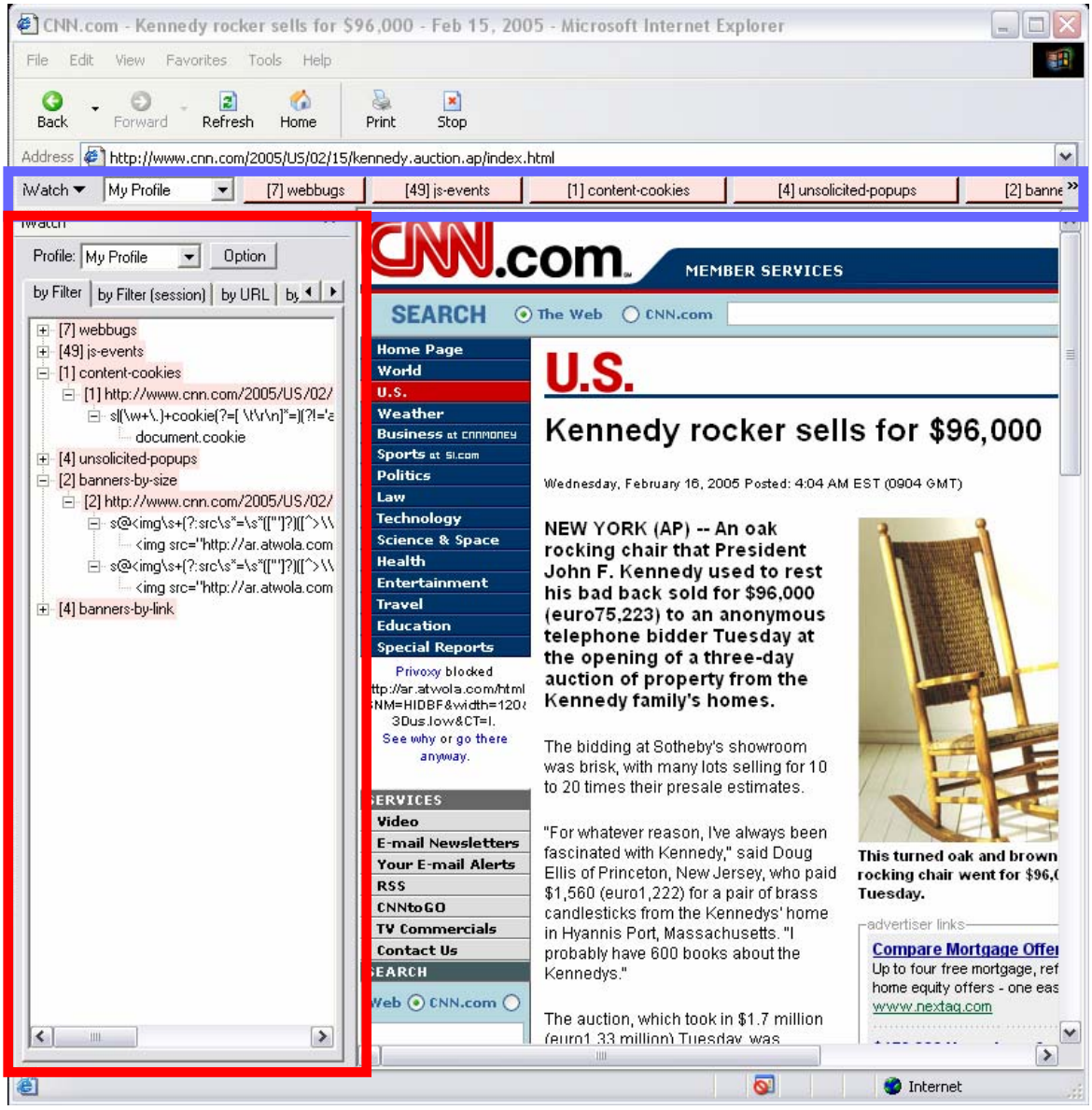


Figure 35: iWatch awareness interface

Two iWatch interfaces running simultaneously in Internet Explorer browser, one marked blue, the other red. The blue interface simply pops us a warning in the toolbar when a threshold is reached, while the red interface shows what got flagged and why.

CHAPTER 6
FRAMEWORK EVALUATION

In chapter three, the five leading frameworks for privacy-aware design were discussed and analyzed in-depth. As part of this analysis, different theoretical advantages and disadvantages were identified for each of these frameworks. This analysis was based on a careful examination of the theoretical foundations on which these were based. Based on this analysis, a new framework was derived, as presented in chapter five. In order to test the correctness of this analysis, and the assumptions on which STRAP is based, it is necessary to see how these frameworks perform in as close to realistic conditions, and by examining real systems.

In this chapter three design experiments are presented aimed at comparing how analysts using different design frameworks perform when analyzing different types of systems. These experiments not only look to determine what kinds of problems the different frameworks help identify, but also how efficiently they do so. One of the principal criticisms of these design frameworks is that there has not been an in-depth evaluation of their effectiveness or usefulness to designers wishing to address privacy issues. These experiments are therefore important because they will help determine what reasonable expectations are for these different frameworks, as well as provide independent validation.

While the three experiments ask analysts to examine systems from domains where privacy has been a significant area of study, no authoritative list of privacy problems exists for any of these domains or systems. It will therefore be impossible to say with any certainty how these frameworks perform in absolute terms. Instead we must determine

how these frameworks perform compared to each other, and whether different frameworks lead to the discovery of different types of problems.

In these experiments we also seek to confirm that people with little background in the area of privacy can use STRAP effectively, at least compared to other methods. This is important because it affects the likely adoption and general utility of these methods. Few organizations are likely to have people on staff specializing in privacy, or be able to afford or prioritize someone with those skills over a more versatile designer. If privacy is to gain weight as a design consideration it is important to not only show that the design methods work, but that the investment required is not prohibitive. While domain experts and privacy experts will in all likelihood perform better than novices, it is important to show whether small teams of inexperienced analysts can produce acceptable results.

In addition to answering an important research question, there are three major advantages or reasons for using novice designers in these experiments. The first is that being at a university this is the most readily available sample populations available to us. The second is that this sample allows us a more direct comparison with the evaluations of Heuristic Evaluation performed by Nielsen & Molich (1990). This is an important point of reference in terms of performance and efficiency when it comes to groups of analysts, and the general validity of the experimental design. As in the case of Nielsen & Molich, these experiments test the analysts' ability to detect problems using these frameworks, not how these frameworks help address these problems. Finally, if novices can show significant performance improvements, effects on experts are less important.

Performing design experiments with real users, using different frameworks is a very time-consuming task, requiring the recruitment of a great many subjects. Because of resource and time limitations, only a few frameworks are evaluated in each of the

experiments. STRAP will always form part of the experiments, as it helps answer some questions about the analysis performed in chapter three, as well as to examine how it performs against the greatest number of frameworks possible. Of the five frameworks examined in chapter three, two were not used in any of the three experiments; the *i** framework (Yu & Cysneiros, 2002), and the framework proposed by Langheinrich (2001). These two frameworks provide the least support to analysts in performing their task (see table 1), with the *i** framework lacking any evaluation criteria, and Langheinrich lacking an analysis method. The *i** framework is also by far the most difficult to apply, and likely the least appropriate for our subjects to use.

As discussed in chapter three, domain can have a tremendous effect on the effectiveness of a design and analysis framework, with two of the frameworks being derived from the study of ubiquitous computing applications. In selecting the systems to be analyzed in these experiments it is therefore necessary to sample both from the domain which these frameworks claim as their own and from others to test how domain-specific these frameworks can be.

The first system selected was Augur, a probabilistic group-calendar system (Tullio et al. 2002). Augur is a web-based, shared calendar that provides additional predictive features intended to facilitate communication within a workgroup. These features include predictions on the attendance of colleagues, as well as information on who has scheduled the same events. These predictions are based on Bayesian networks and improve over time, learning from attendance patterns. With these features, users can identify events that are no longer attended, make informed decisions about which of several conflicting events will be attended, and determine who they will likely see at a particular event.

Users access Augur via secure login. Scheduled events are presented in a standard hour-by-hour, block format. This view is augmented with additional information indicating colleagues who have scheduled the same events and attendance probabilities for those colleagues. Events on a user's calendar have a colored bar to indicate the user's likelihood of attendance as predicted by Augur. Figure 36 shows a set of screenshots from Augur. Augur was chosen because it is representative of a large class of groupware and collaborative systems which have long been studied in terms of privacy.

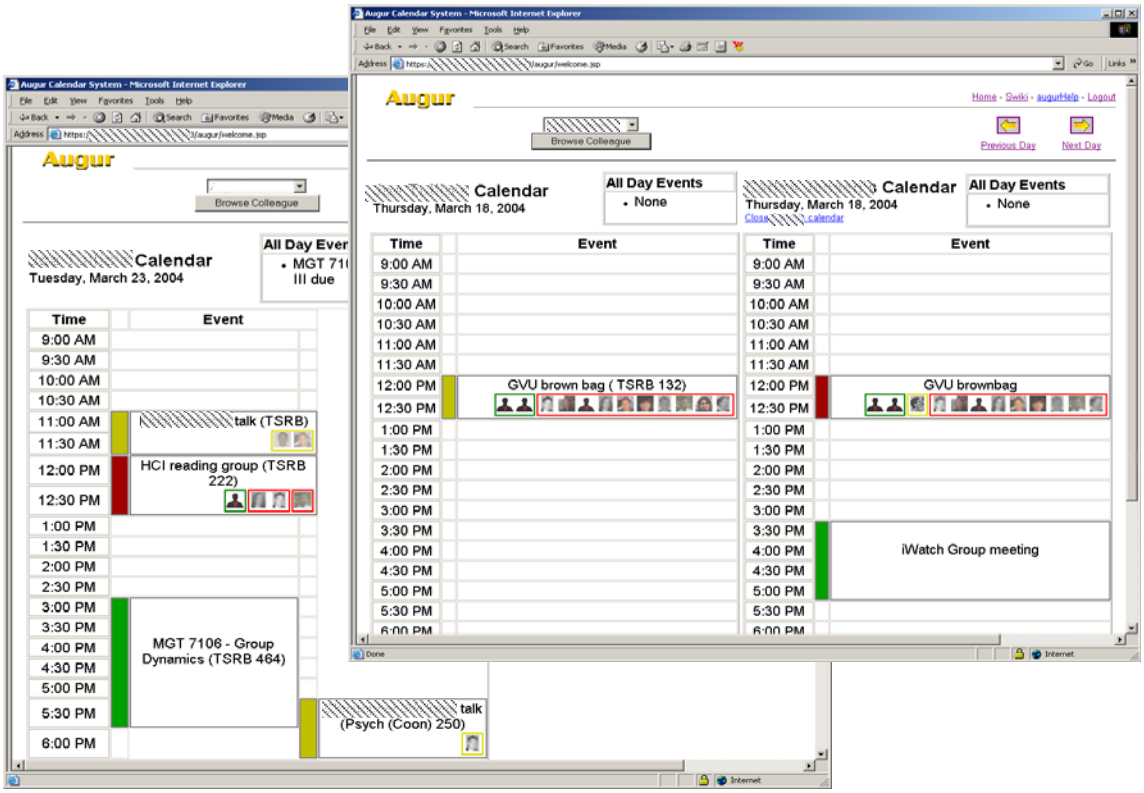


Figure 36: The Augur Group calendar system
Different ways of visualizing a calendar. The color bar associated with an event in the calendar indicates the probability of this person attending (green for likely, red for unlikely, yellow for uncertain). Pictures of other likely attendees, as well as a color-bar to indicate the likelihood of their attendance are also shown

The second system selected was an online bookstore, a representative e-commerce web-site. This system was chosen because it is an example of some of the most common applications with which users interact and disclose personal information with. The choice of system was also influenced by the analysis performed and presented

in chapter four, which allows us to more easily evaluate the quality of the analysis, including whether any significant problems have been overlooked or ignored by any of the frameworks.

The final system needed to represent the ubiquitous computing domain, and closely resemble those systems studied in the derivation of Bellotti & Sellen's framework (1993) and Hong et al.'s Risk Models (2004). The third and final experiment therefore examined a ubiquitous computing environment called Teamspace (Richter et al. 2001), a meeting capture and access system for workgroups. This system is quite similar to the Media Spaces studied by Bellotti & Sellen and Hong et al. in that it employs sensors to capture and make accessible information remotely.

In keeping with Nielsen & Molich's model for their study of Heuristic Evaluation (1990), the subjects for these experiments were students in senior-level classes. In the first two experiments subjects were recruited from senior level HCI classes (CS 4750), with the experiments being performed at the end of the class to maximize their experience and exposure to the problem of design. For the third experiment subjects were recruited from a senior level ethics and professional issues class (CS 4001) were privacy is a major topic, and students spend a fair amount of time studying these types of problems. This change of subject pool was intended to examine the effect that experience in identifying and dealing with privacy issues had on the performance of these frameworks.

In each of these experiments subjects were randomly assigned to one of the analysis methods and given a written description of the method as well as the system. Table 14 shows which frameworks were selected for each experiment. The frameworks were anonymized to avoid bias, though it is impossible to prevent subjects from finding

references to these works or their authors online. The goal was simply to avoid blatant bias rather than prevent a determined subject from discovering the name and origin of each framework. To our knowledge, based on post-experiment debriefings, no subject tried to identify these frameworks on their own, and none of the subjects in any of the experiments had prior knowledge of these methods.

Table 14: Experimental Design

Experimental design showing which frameworks were used for which experiment as well as the number of subjects in each condition and experiment

	Bellotti & Sellen (1993)	Patrick & Kenny (2003)	Risk Models (Hong et al 2004)	STRAP	Total Subjects
Experiment 1: Augur group calendar	15			14	29
Experiment 2: The online bookstore	13		12	14	39
Experiment 3: Teamspace meeting capture and access		5	6	6	17
Total Subjects	28	5	18	34	85

A total of eighty-five subjects took part in the three experiments, as seen in table 14. Subjects in each condition were given printed descriptions of the framework they were to employ. In the case of the five frameworks descried in chapter three this was an anonymized and abbreviated copy of the main paper describing the method. In the case of STRAP this was a condensed version of sections 5.1 and 5.2 without any reference or comparison to other frameworks. In the case of the Bellotti & Sellen framework, an anonymized copy of the Bellotti (1997) paper.

A brief description of the three experiments and their results is given in the following section. A more in-depth analysis of the deeper implications is reserved for chapter seven.

6.1 Augur Experiment

This experiment was completed the summer of 2004, and involved a comparison between the Bellotti & Sellen framework and STRAP. Only two frameworks were selected for this experiment because it was the first in the series and we wished to ensure that we would have enough statistical power to examine the issues we were interested in, as well keeping things simple to ensure that there were no fundamental flaws in our experimental design.

6.1.1 Procedure

For this experiment, 30 undergraduate college students taking an HCI class were recruited to serve as analysts. The students were at the end of the semester-long course, having covered the usual HCI curriculum including heuristic evaluation, GOMS and similar evaluation methods. They had not covered privacy as a specific subject, nor were they familiar with any of the frameworks discussed in chapter three, or STRAP. They had all completed significant project work as part of their class-work (50% of their overall grade).

Subjects were given a description of Augur, a predictive group-calendar system, including a screenshot (Figure 36) (see Appendix A for a copy of the written description given to subjects). They were not given access to the system itself. Subjects were randomly assigned to the Bellotti & Sellen or STRAP condition, with equal numbers in

either. The students were given a 2 hour lecture, roughly half an hour on Augur, the rest of the time was spent reviewing heuristic evaluation and the importance of considering privacy in design. Subjects were given hardcopy descriptions of the method to which they had been assigned.

Subjects were asked to complete their analyses individually, though those in the STRAP condition were allowed to derive goal-tree in groups of up to three students. One third of the subjects took advantage of this opportunity. All subjects returned their results together with an estimated of the time they spent on the analysis. Students were informed that their performance on this experiment would not affect their grade in the class, which had already been set. Subjects were asked to spend at least 60-90 minutes on the analysis.

Two subject-matter experts also performed an analysis of Augur using STRAP. This analysis was used to gauge the performance of the subjects, and the overall efficiency of these methods.

6.1.2 Results

The expert analysis of Augur yielded a total of 36 vulnerabilities, 18 of which were unique (see table 15 for a list of vulnerabilities). This redundancy is a common side-effect of multiple goals requiring access to shared functions or resources, such as databases. We saw multiple examples of such redundancies in the PAB case-study in chapter five involving caching of resources and network communication. Because different analysts may derive different but functionally equivalent goal-trees, this affects how many times an analyst may report the same vulnerability. In order to compare results across conditions and subjects we looked only at the number of unique vulnerabilities reported.

Table 15: Vulnerabilities & Detection Rates - Augur

List of unique privacy vulnerabilities discovered in Augur as well as their count in the expert analysis and detection rates for analysts using STRAP or Bellotti & Sellen

Expert Count	Description	STRAP Detection Rate	Bellotti & Sellen Detection Rate
3	Access logged	**7.1%**	*0.0%*
6	DB/OS: Data encryption	**64.3%**	13.3%
2	Uncertainty in predictions	14.3%	**20.0%**
2	Prediction update schedule/mechanism	**50.0%**	40.0%
1	Event matching errors	**14.3%**	13.3%
1	Event matching schedule/mechanism	**7.1%**	*0.0%*
1	Data freshness/accuracy unknown	**64.3%**	33.3%
8	Communication/Encryption	**50.0%**	6.7%
3	OS/Browser: Information cached, Access/sharing	**35.7%**	*0.0%*
1	Control/awareness over inclusion of image/name/prediction/details	28.6%	**66.7%**
1	No information on who has seen what information about you	28.6%	**33.3%**
1	Third parties may be mentioned in event details	**14.3%**	*0.0%*
1	Reliability of data	7.1%	**13.3%**
1	Automatic synchronization	**28.6%**	13.3%
1	Manually exclude events, Lack of fine-grain control	28.6%	**73.3%**
1	Data parsed to internal format, Loss of context or controls	*0.0%*	**20.0%**
1	Access implies intent	**35.7%**	*0.0%*
1	Automatic login, Revocation/control	**28.6%**	6.7%

The subjects' reports were evaluated and normalized independently by the reviewers. The expert analysts independently marked any false positives, and mapped to the vulnerabilities in the expert analysis. In order to normalize results. Where the reviewers were in disagreement, the subject was given the benefit of the doubt, and the vulnerability was counted.

Of the 30 subjects, 29 completed their assigned analysis (14 in the STRAP condition and 15 in the Bellotti & Sellen condition), and 24 returned data on the time spent on the analysis (13 in the STRAP condition and 11 in the Bellotti & Sellen condition).Table 16 contains an overview of the results.

Table 16: Experimental Results - Augur

Results of experiment in terms of the time spent on task, the average number of vulnerabilities discovered per analyst, and the average number of false positives. Statistically significant results bolded in "difference" row, and standard deviations shown in parenthesis

	Time on task (minutes)	Unfiltered vulnerability reports	Filtered vulnerabilities	General HCI issues
STRAP	88.77 *(25.82)*	6.86 *(2.28)*	5.07 *(2.20)*	1.00 *(0.96)*
Bellotti & Sellen	101.18 *(43.46)*	6.53 *(2.50)*	3.53 *(1.41)*	2.07 *(2.12)*
Difference	+13.49	**-0.33**	**-1.54**	+1.07

There was no significant difference in the total number of reported vulnerabilities (df=27, t= 0.364, n.s.). In the case of the Bellotti and Sellen about one third (31.7%) of the reported vulnerabilities were determined to be general HCI issues (use of colors, placement of information etc.) rather than privacy issues (compared to 14.58% for STRAP). When these were removed from the analysis, we found that there was a

statistically significant difference in number of real privacy vulnerabilities discovered ($df=27$, $t= 2.225$, $p<0.05$), with the subjects in the STRAP condition reporting 43.6% more vulnerabilities than subjects in the Bellotti & Sellen condition.

There were no statistically significant differences in terms of the time spent on the analysis ($df=22$, $t=0.831$, n.s.). This was somewhat surprising because of the additional structure and analysis overhead associated with STRAP. This may indicate that STRAP succeeds in structuring the analysis, lowering the cognitive load by giving the analyst a structure and visual scaffold for the analysis. This in turn means the subjects may have been able to better use their cognitive resources more efficiently in solving the assigned task.

In the STRAP condition subjects discovered a maximum of 9 vulnerabilities, with the average being 5.07 (2.20 stdev), and the median 4. Overall, subjects discovered 17 of the 18 vulnerabilities discovered by the experts (94.4%). In the Bellotti & Sellen condition subjects discovered a maximum of six vulnerabilities with an average of 3.53 avg (1.41 stdev), and the median 3. Overall, subjects discovered 13 of the 18 vulnerabilities discovered by the experts (72.2%).

These results have to be interpreted in light of the information subjects disclosed as part of the post-experiment debriefing, which is that none of the subjects had any experience with group calendar systems. This means that subjects' domain knowledge and experience was very low, a difficult situation for any analysis, and one which made them prone to making false or erroneous assumptions, as well as overlook significant parts of the system simply because of a lack of experience and understanding of the underlying requirements. These are the situations where scaffolding and support is the most important, and we should see a strong effect in favor of STRAP in this case.

Looking at table 15, we note that the Bellotti & Sellen method leads to more cells with a zero-percent detection rate than in STRAP. This suggests design fixation where designers are focusing on a subset of the systems functionality. We cannot however prove this as subjects in the Bellotti & Sellen did not return a system description. Asking them to do so would have forced them to add additional structure their ideas, thereby affecting the results. In the case of STRAP we can turn to more qualitative methods to explore this point, examining the materials subjects returned.

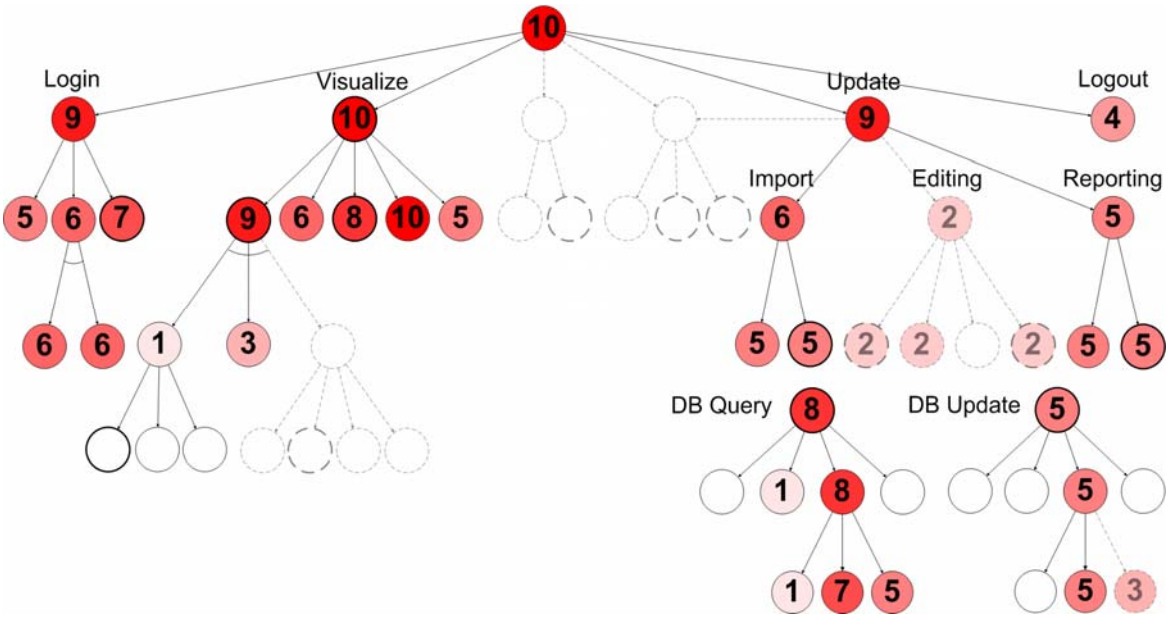


Figure 37: Analysis Quality of Augur Using STRAP

Color intensity and number indicates, on a scale from 1 to 10, the number of subjects identifying these goals as part of their STRAP analysis of Augur

Getting a sense of the subjects' understanding of the system and the process is somewhat straightforward in the STRAP case thanks to the goal trees that students produced. These goal trees can be matched both against the goal tree produced as part of the expert analysis, and those produced by other subjects. An example is shown in Figure 26, where the number inside the node and its color intensity denotes how many diagrams a goal occurred in. Though there were 15 subjects in the STRAP condition, there were

only 10 unique diagrams as ten subjects paired off to make diagrams. A normalized composite of these 10 diagrams is shown in figure 37.

In the first three levels of this decomposition we see that over 50% of subjects identify these goals. The only exception to this rule is the atomic “logout” goal. This suggests that most subjects agreed on and identified the basic functionality of the system. Two diagrams contained a set of editing-related goals (under “Update”). The Augur system does not contain any editing facilities, as stated in the documentation given to students. This was therefore the result of students making incorrect assumptions about the system. These results are compelling in the light of the lack of experience and domain knowledge claimed by subjects.

While we do not have a direct point of comparison in the Bellotti & Sellen case due to the lack of artifacts comparable to the STRAP goal-tree, we can get some sense of what system functionality they considered by what vulnerabilities they reported. By looking at table 15 we get some sense of the issues subjects focused on, in this case primarily awareness and control issues, and what issues they tended to miss, most of the problems associated with infrastructure, such as security of communication, storage, and what the system must do in the background to provide this service. This is of course noisy data because there could be functionality which was considered, but which the framework did not help them identify as being prone to vulnerabilities.

Again, while this is not definitive evidence of design fixation it gives us an indication that subjects in the Bellotti & Sellen case were focusing more narrowly than those in the STRAP case. We also see that the type of analysis and structure imposed by STRAP seems to pay off, at least for problems where subjects have limited domain knowledge, or relatively little experience with this form of analysis. The overall quality

of the diagrams (see figure 38 for a sample) as well as the overall quality and thoroughness of the analysis produced using STRAP (see figure 37) indicates that the effort is worthwhile.

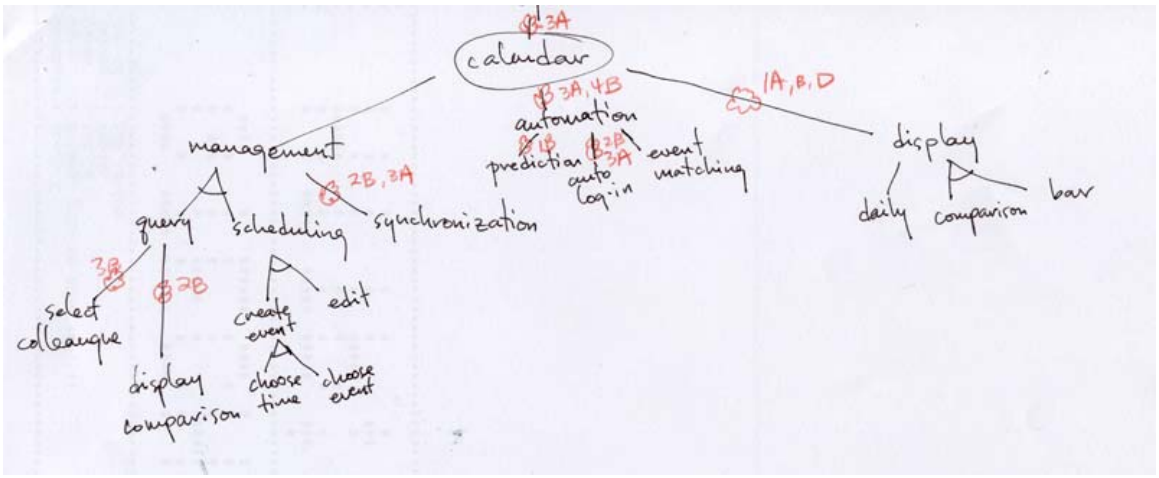


Figure 38: Sample Goal-Decomposition of Augur Using STRAP

Copy of representative diagram resulting from the use of STRAP to analyze Augur. The red numbers and letters refer to a separate indexed list of vulnerabilities

As shown by Nielsen & Molich (1990), heuristic methods (both STRAP and Bellotti & Sellen are in part heuristic based) often lead to more incomplete results. To compensate for this, Nielsen combined the analysis of independent analysts and found that when three analysts combined their results, the results were of equivalent to those produced by more formal methods. We cannot perform an analysis analogous to Nielsen's, because there are no formal methods to use as a baseline. It is however possible to perform a similar analysis by determining the benefits of combining analyses from multiple analysts. This clarifies the costs and benefits of using multiple analysts.

Figure 39 shows how the likelihood of discovering all vulnerabilities increases with the number of independent analysts combining their efforts. The values shown can be considered as an efficiency rating for the method. The efficiency rating in this case is lower than that found by Nielsen for heuristic evaluation. The results are not directly

comparable, of course, because the focus was privacy vulnerabilities, not usability issues in general, and because the heuristics and methods are different. One possible explanation for the difference in efficiency between these methods and Nielsen & Molich's is that these subjects spent, on average, less time and effort on their analysis than Nielsen's subjects did. Subjects were also working on a less familiar domain; none of the subjects used a group calendar tool on a regular basis.

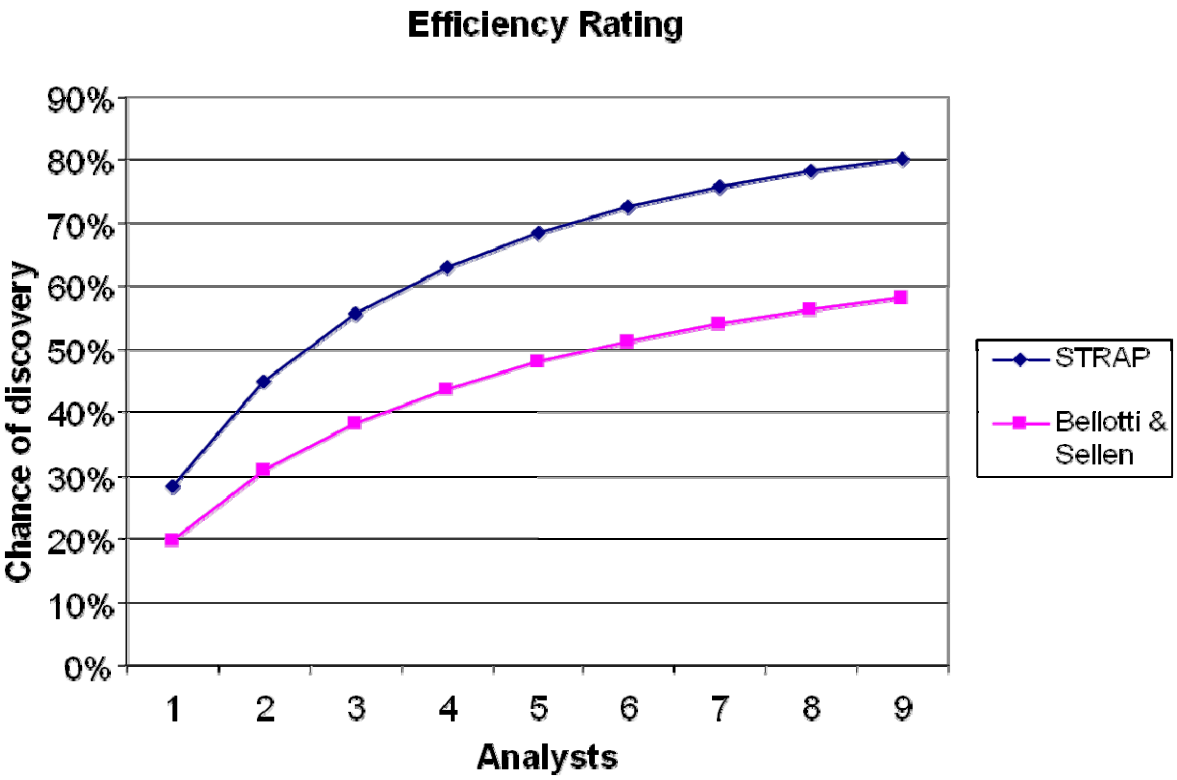


Figure 39: Efficiency of Analysis as Function of the Number of Analysts - Augur

Like Nielsen & Molich (1990), we see a dwindling return on investment with the addition of more analysts or time spent on the analysis, over a certain threshold. How costs and benefits should be traded depends on many unknown factors, such as the average damage caused by an unchecked privacy problem and labor costs. What we can see is that to find 50% of the known problems in this system we would need 2.5 analysts,

or a total investment of 3 hours 42 minutes using STRAP, versus 5.5 analysts, or 9 hours 16 minutes using Bellotti & Sellen's method. If we set the threshold at 60% of known problems, we would need 3.5 analysts, or 5 hours 11 minutes using STRAP, versus 9.5 analysts, or more than 16 hours using the Bellotti & Sellen's method. Given the performance of STRAP and the minimal investment required, it is difficult to argue against its inclusion in the design process.

What was especially striking in this experiment was the level of false positives reported from the Bellotti & Sellen method, and that there was no real difference in the amount of time it took subjects to complete their analysis. The number of false positives in the Bellotti & Sellen case, as well as the "false goals" in the STRAP case, lends weight to the statements subjects made about their level of expertise and experience with these types of systems. When faced with uncertainty, subjects will make a best effort guess. In the case of STRAP this took the form of adding extra functionality to the system, but did not seem to affect the quality of the vulnerabilities identified. In the case of the Bellotti & Sellen case, this may instead have led them to cast their net wider, identifying any potential problem with the system rather than focusing on privacy problems.

While we had expected to see STRAP lead to better results than the less structured Bellotti & Sellen framework, we had expected this gain to come at a price, namely in more time spent on overhead, performing the goal-oriented analysis. It is possible that the less structured, heuristic framework is misleadingly simple. Because analysts do not have a structure to guide the analysis, they waste time determining what to look at next, or keeping track of what has already been considered. This lack of structure also requires analysts to rely on their own cognitive resources rather than external artifacts (e.g. a goal tree), and may also explain the difference in false positives

between the two conditions. Subjects confirmed this in the group debriefing that took place at the end of this experiment.

Overall, this study showed that with relatively little cost and effort, a team of untrained analysts can discover a reasonable number of privacy vulnerabilities. Our expert analysts (one expert in the group-calendar systems, the other in privacy) were equally successful at applying STRAP; each discovering more than 25 vulnerabilities in approximately five hours spent doing the analysis. This shows that STRAP performs well in terms of return on investment (time on task). It is unclear how much of an effect domain knowledge has (either in terms of the domain or in privacy-aware design), but both seem equally important in discovering some of the less obvious problems, as can be expected.

6.2 Online Bookstore Experiment

This study was conducted in the fall of 2004, and involved a comparison between the Bellotti & Sellen, Hong et al.'s Risk Models, and STRAP. The object of this experiment was to compare the heuristics-based frameworks to each other directly, while switching to a different domain. These two frameworks are the most closely related in terms of their structure, the way they have been derived, and the domain they make claims about.

6.2.1 Procedure

For this experiment, subjects from the two sections of an undergraduate HCI class were recruited. The students were at the end of the semester-long course, having covered the usual HCI curriculum including heuristic evaluation, GOMS and similar evaluation

methods. They had not covered privacy as a specific subject, nor were they familiar with any of the frameworks discussed in chapter three, or STRAP. They had all completed significant project work as part of their class-work

In this experiment, subjects were intentionally given less guidance than in the previous experiment. Subjects were asked to not only analyze the core features of a hypothetical online bookstore, but also determine what these should be. They were given a list of basic functionality the system must provide: registering users, login functionality for online sales, the ability to keep track of sales and trends, and otherwise maximize convenience and marketing functions. They were asked not to consider functionality such as end-user book reviews and ratings, but were otherwise allowed to make their own design decisions.

Subjects were randomly assigned to the Bellotti & Sellen, Hong et al., or the STRAP condition in equal numbers. The students were given a 1.5 hour lecture on privacy, and the challenges designers face in addressing privacy issues. The experiment was discussed in class, but the methods were not reviewed, nor were underlying methods like heuristic evaluation or goal-oriented analysis. Subjects were given hardcopy descriptions of the method to which they had been assigned. The papers were anonymized to avoid bias.

Subjects were asked to complete their analyses individually, and return their results with an estimate of the time they spent on the analysis. Students were informed that their performance on this experiment would not be linked to their grade in the class, which had already been set. Subjects were asked to spend at least 90 to 120 minutes on the analysis.

6.2.2 Results

Thirty-nine students returned their assigned analysis (14 in the STRAP condition, 12 in the Hong et al condition, and 13 in the Bellotti & Sellen condition). Of these, 1 student was disqualified in the STRAP and Hong et al. conditions, while 2 students were disqualified in the Bellotti & Sellen condition for not following instructions. In effect these subjects decided not to follow the procedure outlined, or spent less than half an hour on the analysis. In this experiment there was no problem with false positives, possibly because this was a more familiar domain for the subjects to analyze. A summary of the results can be found in table 17.

Table 17: Experimental Results – Online Bookstore

Results of experiment in terms of the time spent on task, the average number of vulnerabilities discovered per analyst, and the average number of vulnerabilities per hour spent on analysis. Standard deviations in parenthesis

	Time on task (minutes)	Vulnerabilities	Vulnerabilities per hour	Observations (discarded / not returned)
STRAP	100.00 *(45.28)*	5.15 *(2.58)*	3.10 *(1.04)*	13 (1)
Hong et al.	102.86 *(38.93)*	4.30 *(2.21)*	2.20 *(0.92)*	11 (1/5)
Bellotti & Sellen	85.71 *(18.37)*	2.60 *(1.17)*	2.07 *(0.78)*	11 (2/3)

There were no statistically significant differences in terms of the time spent on the analysis by the analysts in the three conditions ($F(2,24)=0.45$, n.s.). This indicates that the goal-oriented analysis did not prove a barrier to the student subjects, at least in terms of the time required to perform one. Because this issue was explored in experiment 1, it was not revisited in this experiment.

In the post-experiment debriefing, subjects in the Bellotti & Sellen were asked about their lower time investment compared to the other conditions, and what the causes were. Subjects reported that the Bellotti & Sellen method only outlined so many steps for them to take, and after 90 minutes they had exhausted their options. This finding fits well with the lower correlation between time on task and performance in this experimental condition when compared to the others (see later in this section).

There was a statistically significant difference in the total number of reported vulnerabilities across the three conditions ($F(2,31)=3.70$, $p<0.04$). There were statistically significant differences between the STRAP and Bellotti & Sellen conditions ($df=22$, $t=2.856$, $p<0.01$), and a marginal difference between the Bellotti & Sellen and Hong et al. conditions ($df=20$, $t=1.841$, $p<0.1$). There was no statistically significant difference in the number of vulnerabilities reported in the STRAP and Hong et al. conditions ($df=22$, $t=0.852$, n.s.).

Part of the reason why it is difficult to find a difference between the STRAP and Hong et al. conditions is the variability in the amount of time and effort subjects put into the analysis. While there were no statistically significant differences between the three conditions in terms of average time spent on the analysis, there was significant variance between subjects, as evident in the high standard deviations in both time on task and vulnerabilities discovered. As These two factors are highly correlated ($r = 0.801$), and the three conditions approached significant differences ($F(2,22)=2.93$, $p<0.075$) it made sense to normalize the data and look at the average vulnerabilities discovered per hour.

There was a statistically significant difference in the number of vulnerabilities per hour between the STRAP and Bellotti & Sellen conditions ($df=22$, $t=2.332$, $p<0.05$). There was no statistically significant difference between the Bellotti & Sellen and Hong

et al. conditions (df=20, t=0.620, n.s.). There was no statistically significant difference in the number of vulnerabilities reported in the STRAP and Hong et al. conditions, though there was a trend in favor of the STRAP condition (df=22, t=1.514, p<0.15).

Though there was an overall strong correlation between time on task and the number of vulnerabilities discovered, this was not even across all conditions. Both STRAP and the Hong et al. conditions showed strong correlations between time on task and performance ($r = 0.896$ and $r = 0.743$ respectively). The correlation for the Bellotti & Sellen case is much weaker ($r = 0.568$). This means that of the three methods, STRAP is the one which scales the best in terms of time on task, showing a virtually linear relationship between the two (see figure 40).

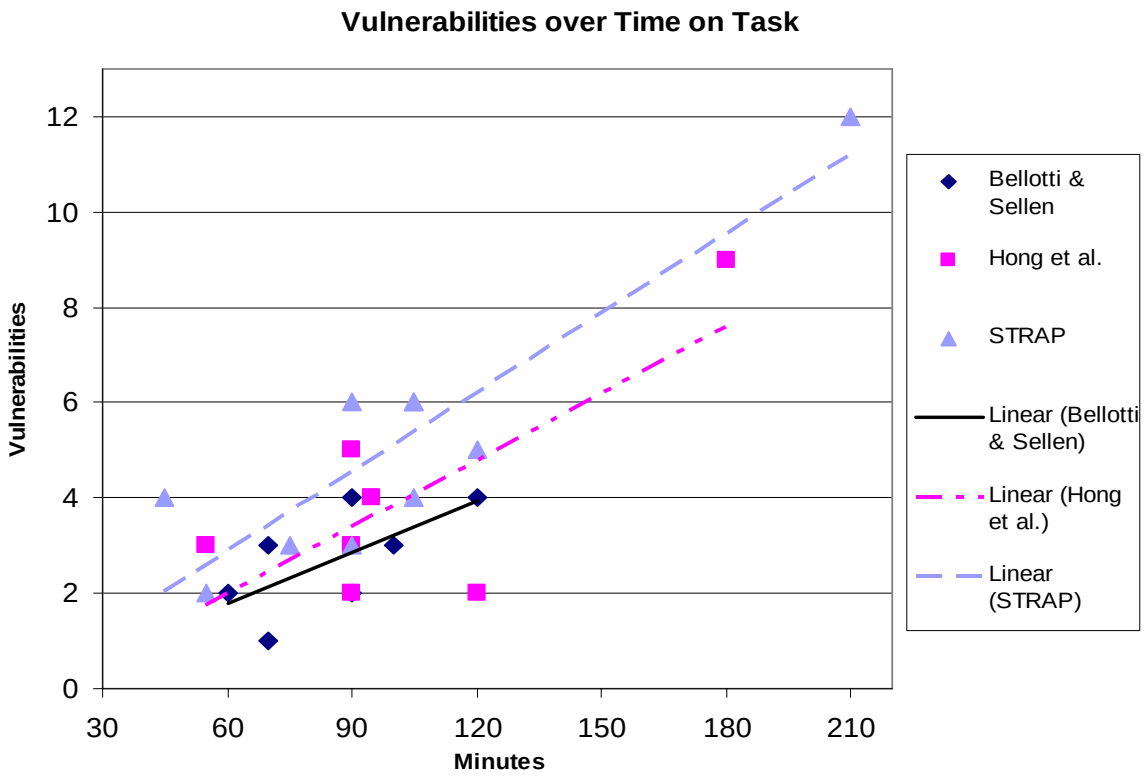


Figure 40: Vulnerabilities Discovered Over Time on Task with Trend-Lines – Online Bookstore

In the STRAP condition subjects discovered a maximum of 12 vulnerabilities, with the average being 5.15 (2.58 stdev), and the median 5. In the Bellotti & Sellen condition subjects discovered a maximum of 5 vulnerabilities, with the average being 2.60 (1.17 stdev), and the median 3. In the Hong et al. condition subjects discovered a maximum of 9 vulnerabilities, with the average being 4.30 (2.21 stdev), median 4.

Overall, subjects discovered all of the 14 vulnerabilities identified based on a refinement of the PAB analysis (see table 18). In the Bellotti & Sellen condition subjects discovered only 11 of these 14 vulnerabilities (78.6%). In the Hong et al. condition 12 of the 14 vulnerabilities were identified (85.7%). In the STRAP condition all 14 types were identified. While the raw number of vulnerabilities discovered was in many cases similar or less than in experiment one, the overall quality of the analysis increased.

In the debriefing, as expected, subjects reported a very high level of familiarity with the domain and the type of system they were asked to analyze. Many reported actually visiting online bookstores as part of this process to more closely examine the mechanisms and practices commonly encountered, as well as the functionality commonly offered. This domain knowledge was undoubtedly an important factor in the overall improvement in the quality of analysis; subjects were routinely able to identify fairly obscure problems. Subjects had a much more detailed and correct model of how the system needed to operate and the steps involved in any operation.

Table 18: Vulnerabilities & Detection Rates – Online Bookstore

List of unique privacy vulnerabilities discovered in the Online Bookstore as well as detection rates for analysts using STRAP, Hong et al., or Bellotti & Sellen Highest detection rate bolded, zero detection rate italicized and highlighted in red.

Description	STRAP Detection Rate	Hong et al. Discovery Rate	Bellotti & Sellen Detection Rate
Hidden parameters	**23.1%**	*0.0%*	18.2%
Policy: incomplete/incorrect	15.4%	9.1%	**27.3%**
Policy: too much work	**46.2%**	18.2%	36.4%
Policy: implications not apparent	**38.5%**	27.3%	18.2%
Policy: Access implies consent	**23.1%**	18.2%	9.1%
Information creep	**46.2%**	36.4%	18.2%
Data use not transparent	**92.3%**	63.6%	36.4%
Cookies: Local caching	**46.2%**	36.4%	27.3%
Pages/Info: Local caching	**15.4%**	9.1%	*0.0%*
Communication: Packet sniffing	46.2%	**54.5%**	18.2%
Communication: logs	23.1%	18.2%	**27.3%**
Database: Encryption	61.5%	**72.7%**	45.5%
Database: Access and management	**23.1%**	9.1%	*0.0%*
Database: Review/access	**7.7%**	*0.0%*	*0.0%*

It is interesting to note in table 18 that despite this fact, there are a number of differences in the types of vulnerabilities the different subject groups identified. While it is difficult to draw any strong conclusions from for instance Hong et al. and Bellotti & Sellen subjects inability to identify the 'Database: Review/access' vulnerability (only one subject in STRAP identified this correctly), we can draw stronger conclusions from the 'Hidden parameters' or 'Database: Access and management' vulnerabilities. These will be explored in more detail in the next chapter.

Figure 41 shows how the likelihood of discovering all vulnerabilities increases with the number of independent analysts combining their efforts. The values shown can be considered as an efficiency rating for the method. These efficiency rating are generally 10 percentage points higher than for Experiment 1.

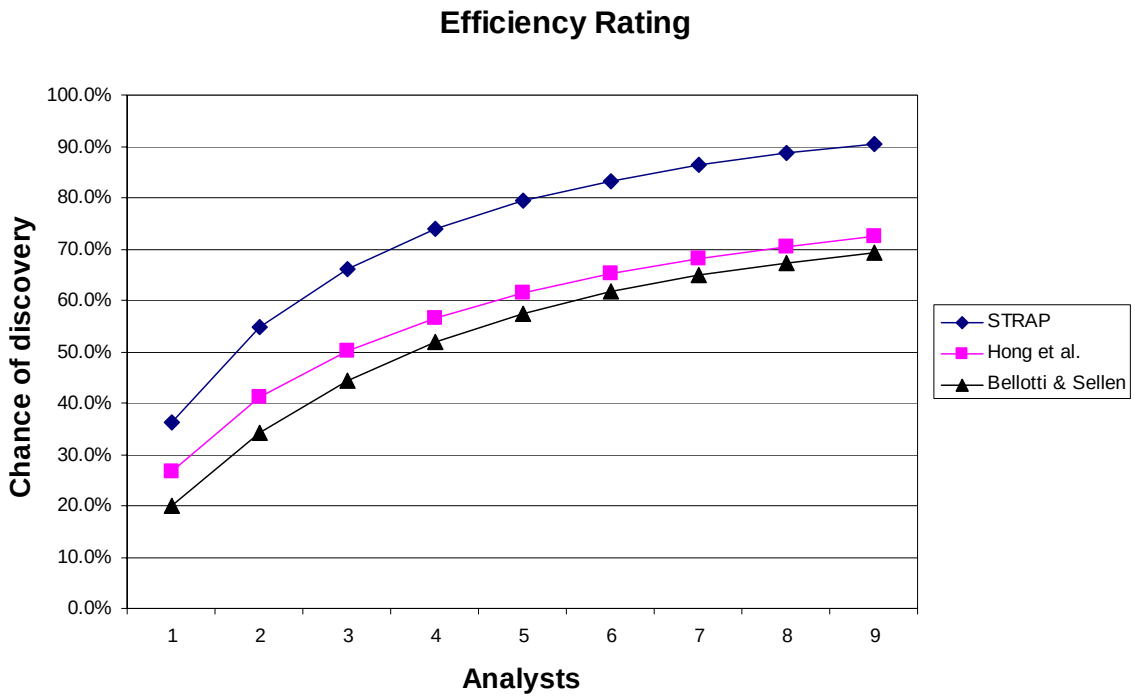


Figure 41: Efficiency of Analysis as a Function of the Number of Analysts – Online Bookstore

As in Experiment 1, we see a dwindling return on investment with the addition of more analysts or time spent on the analysis, over a certain threshold. Where to draw the line between costs and benefits depends on many unknown factors, such as the average damage caused by an unchecked privacy problem and labor costs. What we see is that to find 50% of the known problems in this system we would need 1.75 analysts, or a total investment of 2 hours 55 minutes using STRAP. Using Hong et al. we would need 3 analysts, or 5 hours 9 minutes. Using Bellotti & Sellen's method we would need 3.8 analysts, or 5 hours 26 minutes. If we set the threshold at 70% of known problems, we would need 3.5 analysts, or 5 hours 50 minutes using STRAP. Using Hong et al. we would need 8 analysts, or 13 hours 43 minutes. Using the Bellotti & Sellen's method we would need 9 analysts, or a total of 12 hours 51 minutes.

Overall, we see that the Hong et al. and Bellotti & Sellen methods lead to similar results, including the types of vulnerabilities discovered, and the amount of effort required for either method. Domain experience does seem to have a strong effect on the successful application of these methods, as expected. Unless an analyst is suitably familiar with the application domain, he or she is ill equipped to understand the functionality that needs to be supported, and therefore the small details and hidden catches or assumptions which can cause privacy problems. It is quite possible, from the results of this study, that domain experience is more important than knowledge and experience with privacy problems, as these frameworks seem to be quite effective at helping analysts identify these.

Though the purpose of this experiment was to study the differences in analysis and performance between these three frameworks, the opportunity presented itself to gain some insight into STRAP itself. Instead of following directions, a group of three STRAP

subjects in this experiment decided to skip the goal-oriented analysis, instead performing a much more light-weight hierarchical task decomposition (Shepherd, 1985). They argued that the system they were asked to analyze was so simple to them that they could manage without the more complex and complete breakdown, and simply used the hierarchical-task analysis as a high-level checklist of the functionality they needed to cover (see figure 42 for example of a students' hierarchical task decomposition).

Goals

1. Register Users
    a. Collect Information
    b. Confirm
2. Login
3. Purchase Item(s)
    a. Search for Items
    b. Add Item to Cart
    c. Checkout
4. Poll for Trends
5. Collect E-mails for Newsletter

Figure 42: Sample Hierarchical Task Analysis of Online Bookstore

Copy of representative hierarchical task analysis found in STRAP analysis of the Online Bookstore

While this type of analysis is similar to goal-oriented analysis on a high level, it lacks the level of detail found in a goal-oriented analysis, and it only offers a purely functional view of the system, failing to capture user goals and objectives and how these interact with other agents. Though three observations is too small a sample on which to perform any form of statistics on, it is interesting to note, anecdotally, that using only this rough outline of the system and the heuristics specified in STRAP, these three subjects performed more or less in line with the other STRAP subjects.

One should be careful about drawing conclusions from such a small sample, but these three data points hint at some potentially interesting hypotheses and observations. The first of these is that the relative success of analysts using the STRAP frameworks for analysis is likely not solely due to the goal-oriented analysis. These observations, if generalizable, support the analysis and decisions made in the selection of the STRAP

heuristics. The second observation is that the goal-oriented analysis may be most beneficial for less experienced analysts or those lacking in domain experience and that for more experienced analysts even a very simple structure is sufficient.

Overall it should be noted that the quality of the goal decomposition across all subjects in the STRAP condition taking part in this experiment was very low, with the average subject only identifying 6.0 goals (stdev=3.3), at least four of them being second-level goals. Rather than relying on these diagrams to guide the analysis they were treated as checklists of major system functions, reminders of the functionality to be considered.

6.3 Teamspace Experiment

This study was conducted in the summer of 2005, and involved a comparison between Hong et al.'s Risk Models, the Patrick & Kenny framework, and STRAP. The object of this experiment was to examine how varying degrees of analysis structure help in the detection of vulnerabilities, while switching to an application from the ubiquitous computing domain, similar to those studied in Bellotti & Sellen (1993) and Hong et al. (2004).

6.3.1 Procedure

This third and final experiment largely followed in the footsteps of the other two. To deal with the large number of subjects required, and to further explore this question of how domain knowledge affects performance, subjects were recruited from a required undergraduate computer science class on ethics and social issues in computing. This class spends a significant amount of time discussing and considering privacy issues in software systems. While students do not necessarily have design experience, these classes are

primarily taken by seniors who are likely to already have a broad background in computer science.

These classes received a brief overview of the systems they are supposed to analyze. This lecture did not cover any of the analysis frameworks to avoid biasing the subjects. Subjects were asked to complete the analysis on their own time and return results at a preset date. To ensure subjects took the experiment seriously they received more compensation, in the form of a free lunch, for returning all their materials on time. While fewer subjects opted to participate, the quality of analysis performed by participants was generally better than in previous experiments.

Subjects received a packet containing a paper describing the method they are to use, a description of Teamspace, and the following set of forms to be returned with their analysis:

- A form for reporting the vulnerabilities independently of the rest of the write-up. These forms were anonymized and allowed for a blind review of evaluation of the subjects' performance without bias.
- A time estimate sheet with a breakdown of the different tasks in the assigned method to encourage subjects to give better time estimates.
- A brief survey asking them what their academic major is (CS4001, like CS4750, is sometimes taken by non-CS students), the number of years they have been in the program, and a list of the 3-4000 level (Junior or Senior level) CS courses they have taken. In addition, subjects were asked to rate their familiarity with systems similar to Teamspace. These questions helped determine the kind of potentially relevant experience subjects have going into the experiment.

- A short survey asking them to evaluate the method they employed, what aspects they liked, and what aspects they found confusing or less useful. This survey was the same across all conditions, and was intended to point out areas for future improvements and research.

The complete survey package given to students is included as Appendix B, and the description given of Teamspace is in Appendix C. Subjects were also asked to return any diagrams or other materials generated as part of their analysis. In the STRAP condition this includes the goal-oriented analysis; in the Patrick and Kenny condition this means use cases and object-sequence diagrams.

Six subjects were selected for post-experiment interviews, conducted within days of the return of their written answers. The goal of these interviews was to provide more depth to the quantitative data, gathering impressions and more in-depth evaluations of the process prescribed in each experimental condition. More importantly, these interviews were aimed at explaining the differences in how successful different subjects were in the application of their assigned framework, what parts of these methods subjects are having the most difficulty with, or finding the most useful, and if applicable, why the different methods lead to different results.

Subjects were selectively sampled from the three experimental categories, and included both high and low-achievers. Two subjects were selected from each experimental condition, and each interview lasted approximately thirty minutes. These interviews were loosely structured around the questions in the questionnaire and their analysis.

Subjects were asked to expand on their academic and professional background, especially any experience they might have in systems design or analysis. Subjects were

asked about their experience with systems such as Teamspace, their understanding of how Teamspace works, how the framework they were assigned works, and the analysis they performed. For this final task, subjects were given a copy of their analysis and asked to walk me through their analysis, explain the decisions they had made, as well as any doubts or insecurities they had about the process. The goal was to as closely as possible reconstruct their thought process at the time of the analysis. This was the primary reason for scheduling the interviews immediately after the analysis.

6.3.2 Results

Despite a fairly intensive recruitment effort in which a large number of subjects volunteered, only a small number of subjects actually completed the experiment. It is not entirely clear why this is the case. It may be that because this was not a design-oriented class, subjects were less interested or used to such exercises. It may be that the pressures of the CS4001 class towards the end of the semester are more intense than in the CS4750 class, and people are therefore less likely to take time out of their schedule to complete a voluntary assignment.

While this experiment took place as part of an international study-abroad program, the location and the context are unlikely to have significantly affected recruiting, given that Experiment 1 was carried out in the same context a year earlier. The most significant difference between these two experiments was that a different class was sampled.

Only seventeen subjects completed the task as prescribed, though through incentives (free lunch), the overall time and effort spent on the assignment increased. Of

the seventeen subjects, six completed the STRAP assignment, six completed the Hong et al. assignment, and five completed the Patrick and Kenny assignment.

The relevant and interesting findings from the survey package are given in table 19, with most of the rest of the questions proving to be relatively meaningless for the purposes of analysis. Subjects indicated very low levels of expertise with the different types of related analysis techniques, the median on a 5-point Likert scale being 1 (never heard of it) for goal-oriented analysis, 2 (some familiarity) for Heuristic Evaluation, 2.5 for use cases (between some familiarity and have used it), and 3 (have used it) for object-sequence diagrams. Of the application domain, subjects rated their expertise as 1 (never heard of it), though many cited systems such as instant messaging as examples.

Table 19: Experimental Results – Teamspace

Results of experiment in terms of the time spent on task, the average number of vulnerabilities discovered per analyst, and the average number of vulnerabilities per hour spent on analysis. Standard deviations in parenthesis

	Time on task (minutes)	Time on structuring (minutes)	Vulnerabilities	Vulnerabilities per hour	Observations	Non-CS Students	Years as Students
STRAP	161.5 *(70.2)*	55 (30.7)	4.50 *(1.22)*	1.75 *(0.96)*	6	2	3.1 *(1.11)*
Hong et al.	128.00 *(52.1)*		2.33 *(1.21)*	1.77 *(0.86)*	6	1	2.5 *(1.00)*
Patrick & Kenny	135.00 *(34.4)*	58 (25.0)	2.60 *(1.14)*	1.27 *(0.85)*	5	2	3.40 *(0.95)*

Subjects across all three conditions gave relatively high marks to the frameworks and the clarity of the task they had been assigned, with median scores on a 5 point Likert scale being a 4 on the clarity of the Teamspace documentation and the framework

description. Subjects had high confidence in their understanding of the frameworks at the end of the process (4), with Hong et al. subjects seemed being the most confident (5) and Patrick & Kenny the least (3). The Patrick & Kenny framework was again seen as the most difficult to learn (3) and use (2) compared to the other two (4 on both frameworks in both questions). This affected subjects' perception of the usefulness of the method (2 for Patrick & Kenny, 4 for Hong et al., 3 for STRAP). Regardless, subjects thought the application of the frameworks to be a worthwhile exercise (4).

On average, subjects reported spending over two and a half hours on the analysis part of the assignment. This is significantly longer than what the assignment asked them to do (90-120 minutes) and with the two previous experiments (95 minutes in experiment one and 96 minutes in experiment two). There were no statistically significant differences in terms of the time analysts in the different conditions spent on their analysis ($F(2,14)=0.62$, n.s.) Subjects in the STRAP and Patrick & Kenny conditions spent on average the same amount of time structuring the problem (performing goal-oriented analysis or deriving use-cases and object-sequence diagrams), approximately 40% of their total time on task in this experiment.

In the STRAP and Patrick & Kenny conditions, two subjects in each condition reported being non-CS students, while only one student in the Hong et al. condition reported being a non-CS student. Non-CS students majored in Electrical Engineering, Architecture (2), Electrical and Computer Engineering, or Management of Information Systems. Subjects in the Hong et al. condition on average reported having been in school for less time than students in the other two conditions, though these differences were not statistically significant ($F(2,11)=0.80$, n.s.).

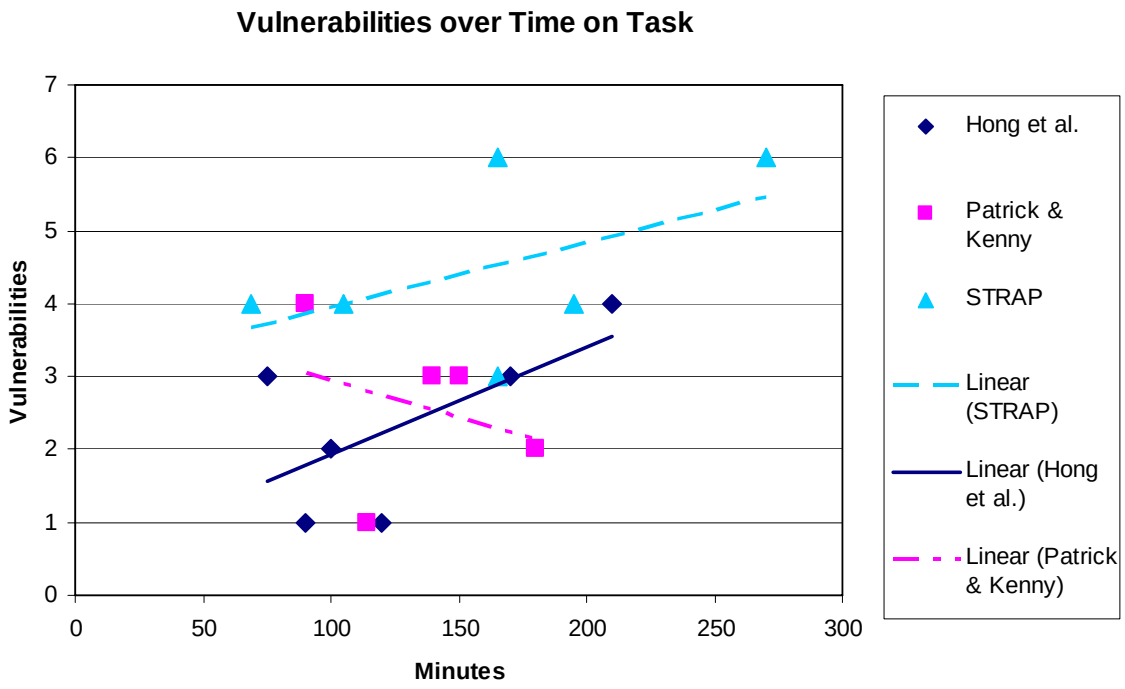
The effects of these differences were negligible. Across conditions, there were no statistically significant differences between CS and non-CS majors in terms of the number of vulnerabilities discovered (df=16, t=0.7027, n.s.), vulnerabilities discovered as a function of time-on-task (df=16, t=0.6528, n.s.), or in the amount of time subjects spent on their analysis (df=16, t=0.1173, n.s.). Furthermore, there was no correlation between the length of studies and the number of problems discovered (r = 0.0249), and only very weak relationships between the length of study and the number of vulnerabilities discovered per hour spent on the analysis (r = 0.4788) and a very weak negative relationship between length of study and time spent on analysis (r = -0.4479).

There was a statistically significant difference in the total number of reported vulnerabilities between the different groups ($F(2,14)=5.74$, $p<0.02$). On closer examination it was determined that there was a statistically significant difference between the STRAP and Patrick & Kenny conditions (df=10, t=2.602, $p<0.05$), and between the STRAP and Hong et al. conditions (df=11, t=3.0219, $p<0.05$). There was no statistically significant difference in the number of vulnerabilities reported in the Patrick & Kenny and Hong et al. conditions (df=10, t=0.374, n.s.).

The correlation between time on task and vulnerability discovery was much weaker than experiment two (r = 0.4754 compared with r = 0.801). Though subjects in the STRAP condition reported more vulnerabilities they also spent more time on their analysis on average. There were no statistically significant differences in the number of vulnerabilities subjects reported per hour of analysis using the three frameworks ($F(2,14)=1.56$, n.s.).

There were weak correlations between time on task and the number of vulnerabilities reported in the STRAP and Hong et al. conditions (r = 0.5126 and

r = 0.6336 respectively). In the case of the Patrick & Kenny there is no linear relationship (r = -0.3082). Figure 43 shows what this means in practical terms. As we can see, there is little evidence of a linear relationship between time on task and vulnerabilities in the Patrick & Kenny condition, and what there is is negative. The linear relationships in the



STRAP and Hong et al cases are stronger, though still weak.

Figure 43: Vulnerabilities Discovered Over Time on Task with Trend-Lines – Teamspace

It is interesting to note that the Hong et al correlation is stronger than that in the STRAP condition. This means that if the expected trend-lines were extended infinitely, subjects in the Hong et al. condition would overtake those in the STRAP condition at the 7 hours and 20 minutes mark. At that point, analysts using either framework would report, on average, seven vulnerabilities each. This estimation requires us to predict performance with only a single observation over five hours, and these extrapolations should therefore be made with care.

A total of eleven unique vulnerabilities were discovered by analysts (see table 20). In the STRAP condition subjects uncovered all eleven vulnerabilities, with each analyst on average reporting 4.5 vulnerabilities (1.22 stdev), and the median 4. In the Patrick & Kenny condition subjects discovered 6 (54.5%) unique vulnerabilities, with the average being 2.60 (1.14 stdev), and the median 3. In the Hong et al. condition subjects discovered 7 (63.6%) unique vulnerabilities, with the average being 2.32 (1.21 stdev), and the median 2.5.

Table 20: Vulnerabilities & Detection Rates – Teamspace

List of unique privacy vulnerabilities discovered in Teamspace as well as detection rates for analysts using STRAP, Hong et al., or Patrick & Kenny. Highest detection rate bolded, zero detection rate highlighted in red.

Description	STRAP Detection Rate	Hong et al. Discovery Rate	Patrick & Kenny
Security: Unauthorized access	**100.00%**	16.67%	**100.00%**
Control: Lack of editing and low-level access control tools	**50.00%**	**50.00%**	40.00%
Control/Awareness: Events/Information taken out of context	50.00%	**66.67%**	*0.00%*
Notice/Awareness: Lack of awareness of data use	16.67%	33.33%	**40.00%**
Choice/Consent: Forced use of system (corporate policy)	16.67%	33.33%	**40.00%**
Security: Transmission of data to and from clients	**50.00%**	*0.00%*	20.00%
Awareness: Unknown consequences of actions	16.67%	16.67%	**20.00%**
Notice/Awareness: Lack of awareness/reminder of capture	**50.00%**	16.67%	*0.00%*
Security/Consent: Calendar access/sharing for meeting scheduling	**50.00%**	*0.00%*	*0.00%*
Security: Data storage safeguards on server	**33.33%**	*0.00%*	*0.00%*
Choice/Consent: Impossible to retract consent	**16.67%**	*0.00%*	*0.00%*

As we see in table 20, and as we saw in the previous two experiments, definitive patterns emerge in terms of the kinds of vulnerabilities subjects in the different conditions are able to detect and identify. We see significant overlap between the Hong et al. framework and the Patrick & Kenny frameworks in terms of weakness in identifying security-related vulnerabilities, whereas STRAP appears to be somewhat weaker on

identifying the social issues surrounding the use of information. This again is an issue which will be examined in greater detail in the next chapter, though some aspects are important to bring out at this point.

Of the five frameworks reviewed in chapter three, the Patrick & Kenny framework is the one most similar to STRAP with its use of use-cases and object-sequence diagrams to structure the analysis. It is therefore surprising to see how differently it performs to STRAP, how similarly it performs to Hong et al. (even underperforming in some cases), and how poorly it was received by the subjects. This was especially surprising given that the system description was presented in scenario-style (see Appendix C), and should therefore have lent itself naturally to the kind of analysis Patrick & Kenny advocate.

Generally speaking, subjects on average returned one short scenario, and one very basic object-sequence diagram per analysis. This is a far shallower analysis than what Patrick & Kenny advocate, and what subjects in the STRAP condition performed. Teamspace was in a domain most subjects had some familiarity with, but few were expert in. In terms of subjects' domain experience it likely falls somewhere between Augur, where subjects benefited from the structured analysis, and the Online Bookstore, where the benefits of the structure were perhaps less clear. Subjects in the Patrick & Kenny condition should therefore have performed better than those in the Hong et al. condition. To explore this issue we turned to the post-experiment interviews where we asked the two subjects in this condition to explain this behavior. To protect the identity of the subjects all their names have been altered in this dissertation. This applies not only to this section, but the discussion in chapter seven.

Matt was a senior who claimed to have extensive experience in software engineering methods such as use-cases and UML from both class-work and some outside projects, and had the course background to back these claims. Matt liked the idea of using something similar to UML, but had reservations about the object-sequence diagrams: "*I've been on design projects before, and, I would do UML, sure, but use cases and object sequence diagrams are usually stopped at, um, keeping it in my head, I've never taken that last step of fleshing it out and writing it out except for projects where that was part of the grade.*" In this experiment Matt described a very basic scenario (setting up a meeting), and did an object-sequence diagram to describe it.

As to why subjects did not do a more thorough job of specifying use cases and object-sequence diagrams, Matt felt that "*if you haven't done it already* [as part of the other design steps] *then it's because it's common sense, or its fairly easy and you can already grasp it in your head.*" In other words, Matt felt that breaking things down into that level of detail was too much of a burden and therefore unnecessary. Given the overall results of this analysis, it is not clear if Matt was right, but he does bring up an important point. If this analysis had been performed as part of another design step, or could have been used for other purposes, then the costs of performing this analysis would likely be more justifiable. Performing such a detailed and time-consuming analysis for this purpose alone may be excessive. Matt did however claim to spend two and a half hours on the analysis and discovered three vulnerabilities (above average on both counts).

Erik was slightly less dedicated to the analysis than Matt, spending just under two hours. Like Matt, he did a simple object-sequence diagram of one of the scenarios in the system description, but stopped half-way and instead worked off the list of heuristics

Patrick and Kenny presented. Like Matt, Erik felt that the burden of performing such a detailed analysis, a description of every major use-case, and a complete object-sequence diagram of all the interactions in that scenario was overwhelming and unnecessary, and that it was a likely reason for why some subjects had decided to drop out after being assigned to this experimental condition.

Erik only identified one vulnerability; the risk of unauthorized access to data stored on a central server. Erik explained his frustration with the method: "*My biggest problem with this was, I think it's called Pisa* [one of the steps in the Patrick & Kenny method], *the actual process, and it goes through… the HCI requirements are necessary to meet like the legislation and stuff, and the Teamspace description doesn't talk about that at all. So I was like… I mean, there is nothing in the description that this is what we're going to do, so I was like… So this really doesn't apply to the Teamspace project, or at least the Teamspace project didn't address those issues yet, so that was the biggest thing.*"

Patrick and Kenny do build their framework around meeting legal requirements through the use of adequate HCI design. Erik could not get past this notion, and spent all this time trying to determine how Teamspace's interface could or did address privacy issues, and what legal requirements Teamspace might be under. While this later point is somewhat unique to Erik, the first behavior seems prevalent in the kinds of descriptions the other subjects in this condition gave of the vulnerabilities they described. While Teamspace, as described in chapter three, can be considered a fairly general method, subjects seemed to get hung up on just the interface requirements and legal jurisdiction.

Returning to the issue of relative performance, figure 44 shows how the likelihood of discovering all vulnerabilities increases with the number of independent

analysts combining their efforts. The values shown can be considered as an efficiency rating for the method. The efficiency rating for the STRAP and Hong et al. case are in line with the results from experiment two, and the Patrick & Kenny shows similar results to the Hong et al. case.

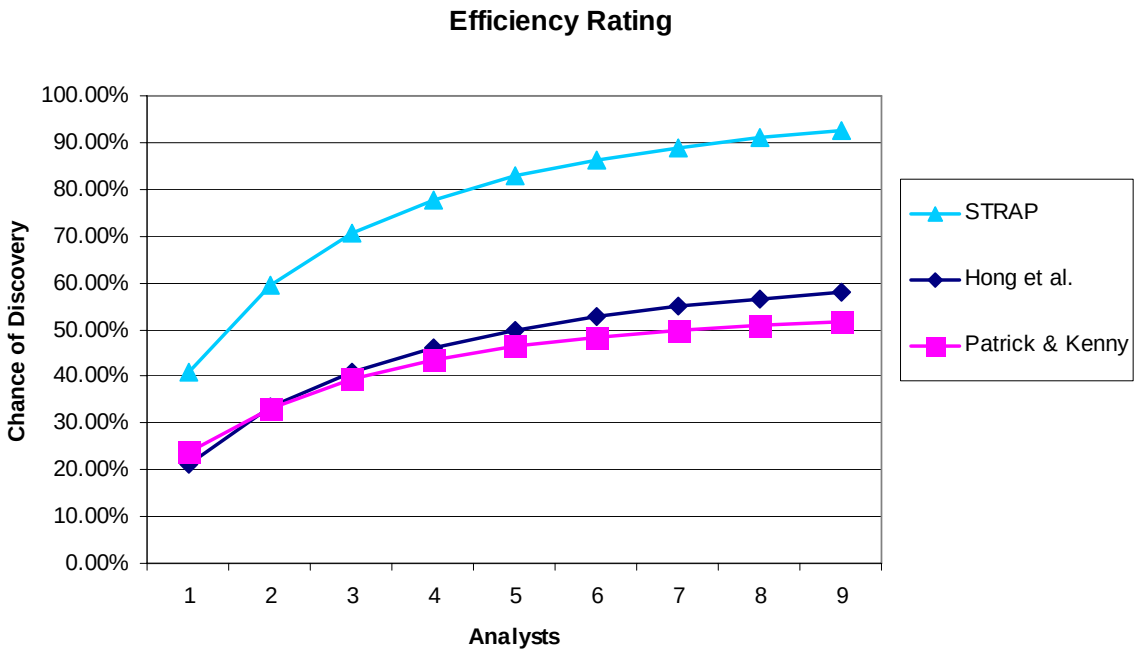


Figure 44: Efficiency of Analysis as Function of the Number of Analysts – Teamspace

As in the previous two experiments, we see a dwindling return on investment with the addition of more analysts or time spent on the analysis, over a certain threshold. One interesting thing to note in this experiment is that the curves for the Patrick and Kenny framework and the Hong et al. framework cross each other at the two analyst mark. This is because though the analysts using the Patrick and Kenny analysis had a greater chance of discovering the problems they could detect, they detected fewer types of problems than those using the Hong et al. framework (see table 20). This is possibly an artifact of the low number of analysts, especially when compared to the STRAP condition.

In terms of overall efficiency we see that to find 50% of the known problems in Teamspace we would need 1.5 analysts, or a time investment of 4 hours 2 minutes, using STRAP. Using Hong et al. we would need 5 analysts, or 10 hours 40 minutes. Using the Patrick & Kenny framework we would need 7 analysts, or 15 hours 45 minutes. Setting a higher threshold is meaningless, as neither Hong et al. nor Patrick & Kenny achieve much higher than a 50% success rate.

CHAPTER 7
DISCUSSION

In the previous chapter I presented three experiments comparing and contrasting the relative performance of subjects using four different design frameworks aimed at identifying and resolving privacy issues in the system design phase. These experiments compared a selection of frameworks to each other in different settings and application domains, with the goal of determining how analysts using the different frameworks performed, how many problems they identified, and what differences there were in the analysis. For the sake of simplicity, these three studies were presented in chronological order, and results largely presented separately for each experiment.

The goal of this chapter is to take a step back and re-examine not only the data presented in chapter six but the results from the post-experiment interviews of experiment three, and also the studies presented in chapter four. The goal is to examine the implications and limitations of the studies and the frameworks, including STRAP, and for privacy-aware design in general. This analysis will be done with a special focus on answering the questions posed in the thesis statement, and others identified in the course of this thesis, each of which will be discussed in their own sub-section.

The main questions examined are the effect of analysts' expertise on the efficiency of analysis, the scalability and adaptability of these analysis frameworks, and finally revisiting the design decisions for each of these frameworks and their appropriateness. This includes looking at the types of issues these frameworks help analysts discover, which they tend to overlook, and why.

7.1 Effects of Knowledge and Experience

Experience is usually an important indicator of success, as subjects learn and develop more efficient and effective work habits, spend less time figuring out what to do, and generally make fewer mistakes. This is of course not unique to privacy-aware design or these frameworks. The important question here therefore is to determine what kind of experience makes a difference, and what effect it has on performance. In this situation there are a number of different types of expertise which could have an effect: expertise with the frameworks themselves, expertise or knowledge of privacy issues, expertise and knowledge of the domain and the class of systems being analyzed, and finally more general experience as a designer and computing professional.

All of these types of experience could potentially be important in determining how successful a given designer will be at detecting and addressing privacy issues, yet it will be relatively difficult to find someone who meets all these requirements. As such, it is important to identify which of these traits are most desirable and important, or what type of experience one should focus on developing to be most successful. This issue was explored in some detail in the third experiment, where subjects completed a survey detailing their level of familiarity and expertise. We can also draw on less structured data from experiments one and two, specifically post-experiment debriefing sessions asking how familiar subjects were with the type of system they were asked to analyze.

In the analysis of experiment three (chapter 6.3) we determined that a background in computer science does not affect the outcome of analysis. All subjects had a technical background, but only 12 of the 17 (70.6%) subjects had a background in computer science. We showed that there was no statistically significant difference in performance or the effectiveness of analysis between subjects with a CS background and those

without. Furthermore, there was no strong interaction between the length of study and the results of analysis. There were also no significant differences or interactions between performance and familiarity with goal-oriented analysis, heuristic evaluation, use cases, object-sequence diagrams, or the application domain. This being said, on average, the level of experience and knowledge was very low (average of 2 across all subjects and categories on a 5 point Likert scale, with only 3.3% of responses having a value of 4 or higher).

While the above seems to indicate that academic experience had little impact on the performance of subjects, it is important to remember that we are dealing with subjects with very little academic and practical experience. This was an intentional choice to determine if these methods had a low enough threshold, but it may have been too low a threshold to determine what kind of effect experience has on performance. It is also true that none of the students in the three experiments had any previous experience looking at privacy issues.

As a counterpoint to this, the expert STRAP analysis of Augur (Experiment 1, section 6.1) was performed by the developer of Augur, an expert in group calendar systems, and myself, experienced in privacy analysis and expert in the application of STRAP. As noted in table 16, a total of 36 vulnerabilities were discovered as part of this expert analysis. The domain expert and I spent approximately the same time performing the analysis independently, three hours, and each discovered twenty-five privacy issues. The domain expert also identified a number of false-positives, eight in total, three related to human error and general HCI requirements, and five related to database access. These were not serious errors, just an indication of the eagerness and relative inexperience with

the method of analysis. Despite these missteps, the domain expert identified ten vulnerabilities which I failed to identify, and vice versa.

The type of vulnerabilities we identified were different, with my unique discoveries being related to the underlying structure and mechanics of the system; the transportation and storage of information across the network, as well as issues inherent in the web browser users used to access the system. The domain expert had a more in-depth knowledge of how calendar systems work, the opportunities these offer for sharing data, and the information that can be gleamed from this data. As a consequence, I discovered more security and implementation dependent vulnerabilities, while the domain expert identified more awareness, social, and organizational problems.

Looking at the results of the three experiments and the types of issues untrained subjects discovered in each, we see that subjects tended to miss the more obscure problems. By obscure, here we mean problems related to some subtlety of the systems functionality that they missed or failed to appreciate, or to some technical peculiarity or consequence of the systems infrastructure. While this is a very high-level generalization, the results in chapter six and the analysis in section 7.3 support this claim.

While the results from the expert analysis are mostly anecdotal, and it is impossible to draw conclusions from such a small and potentially biased data-set, it does point to some interesting hypotheses. It seems that both types of expertise are important, and that both are needed to detect a wide set of problems. It also seems that as expected, expertise, whether of privacy or the application domain, has a very strong positive effect on the number of vulnerabilities discovered. Subjects across the three conditions discovered on average five vulnerabilities in an hour and a half, the experts in experiment one discovered on average twenty-five vulnerabilities just over three hours. This

difference in performance between experts and novices is likely in the application of the other frameworks as well, though exactly how strong an effect is unknown. This has positive potential for the effectiveness of semi-skilled analysts working in teams, and may boost performance up to the levels seen in Heuristic evaluation.

Another of the original hypotheses was that these frameworks, and in particular STRAP, would be efficient and effective in the hands of relatively inexperienced analysts. The three experiments did demonstrate that though individuals could have relatively low success-rates, groups of analysts could be highly efficient at detecting problems. This was especially true in the case of STRAP, which was much more efficient than the other frameworks, both for individual analysts and groups.

One conclusion from these experiments is that if designers train themselves to think about privacy issues informally, system requirements and designs would likely include fewer privacy vulnerabilities. However, the adoption of more systematic, but still light-weight analysis frameworks such as STRAP and Hong et al. has dramatic effects of design thinking and the number of privacy problems identified.

It should be noted that though the relative differences we found in performance between these frameworks should be an accurate reflection of their usefulness to users, care should be taken in reading too much into the absolute percentages of vulnerabilities discovered. These percentages are based on the number of vulnerabilities known to the author and other experts I have collaborated with, and may not be an exhaustive or complete list. We cannot say with absolute certainty that these are all the vulnerabilities in any of these systems, something which may change the percentages. Regardless of the actual number of vulnerabilities we may have overlooked, these will affect the success rates of the three conditions equally.

It should also be noted that though the expert analysis was always performed using STRAP. This could have been a source for bias had the other expert and I not followed the procedures laid out by Nielsen and Molich (1990) of amending the master list of vulnerabilities and goals-trees with anything subjects discovered which we overlooked. This only occurred once, in experiment one, where subjects identified that the system needed to support a logout function, something we had overlooked. Thankfully this goal did not lead to any unique vulnerabilities, and illustrates how even experts can miss.

7.2 Scalability and Adaptability

Closely related to this issue of expertise, and in particular domain knowledge, is the question of whether the domain makes a difference in the application of these frameworks. The frameworks described in chapter three were developed with either a particular type of application in mind (Bellotti & Sellen and Hong et al.), or to address a particular set of privacy issues (legal requirements; Patrick & Kenny), which effectively defines a domain, a class of applications which primarily encounter these types of problems. These design decisions are important because they can affect the applicability of a framework, and the value of learning it. The three experiments and the systems analyzed were carefully designed to examine this issue.

Looking at the relative performance of analysts using these different frameworks in the three experiments, we see little evidence to support the original hypothesis that STRAP would be more generally applicable. We had especially expected those frameworks based on heuristics derived from the study and observation of existing systems (Bellotti & Sellen and Hong et al.'s Risk Models) to be the most limited in terms

of applicability, something which is not supported by the experimental data. Table 21 shows that the Hong et al. and Bellotti & Sellen frameworks performed reasonably well across conditions. In fact, subjects on the Risk Models condition performed worse in Experiment 3 than in the other two. This may have been caused by the change in experimental population, with subjects in this condition being more reliant on knowledge of HCI concepts than subjects in the STRAP condition.

Table 21: Experimental Results – Average Performance

Percentage of problems discovered as a group, with average percentage of vulnerabilities discovered per subject in parenthesis. All averages weighted by number of subjects

	Bellotti & Sellen	Patrick & Kenny	Risk Models	STRAP	Weighted Average
Experiment 1: Augur group calendar	72.2% *(19.6%)*			94.4% *(28.2%)*	83.0% *(23.7%)*
Experiment 2: The online bookstore	78.6% *(18.6%)*		85.7% *(30.7%)*	100% *(36.8%)*	88.5% *(28.8%)*
Experiment 3: Teamspace meeting capture and access		54.4% *(23.6%)*	63.3% *(21.8%)*	100% *(40.9%)*	73.8% *(28.9%)*
Weighted Average	75.2% *(19.1%)*	54.5% *(23.6%)*	78.4% *(27.5%)*	97.7% *(34.0%)*	83.6% *(27.1%)*

While these three experiments asked analysts to look at different types of systems, there was a common thread to these systems, including the fact that all somehow involved the use of a web-browser. Despite this fact, these three systems do represent a fairly diverse and interesting set of applications, demonstrating the value of applying these types of frameworks in real life.

Though performance was limited in the case of inexperienced analysts, the investment required to use any of these frameworks, with the possible exception of the

Patrick & Kenny framework, was so low as to still make it worthwhile. More encouraging, the analysis by the developer of Augur shows that just domain knowledge and motivation can lead to a very successful analysis for a very minimal investment in learning to apply one of these frameworks.

Of the two types of experience, based on the post-experimental interviews, it seems subjects were most concerned about their experience with privacy problems and analysis. One of the subjects from the STRAP condition, Stacy put it best. "*I had a hard time just trying to find… the security issues that I found weren't the ones like 'operating system…' what I found was more things like 'we're all working together and you can check when I'm there and when I'm not here.* […] *I feel like if you took someone like maybe* [name of subject with more technical expertise], *and gave him this system to analyze and maybe he'll go more into the technical side.*" This was not an uncommon sentiment, and subjects seemed most insecure about these types of vulnerabilities. Stacy did mostly focus on more social and organizational types of vulnerabilities in her analysis, so her self-assessment was honest.

Subjects also recognized that domain knowledge and past experience played an important role in the quality of analysis, and the types of vulnerabilities discovered. One subject, Mark, from the Hong condition said: "*I think it* [the results of the analysis] *will depend on who you ask, I think you'll get a wide spectrum of issues and results, I think you bring your own perspective, and I don't think these questions* [heuristics] *change that*." Stacy from the STRAP condition agreed. "*I think different people will bring different perspectives to the analysis, different focus, and discover different vulnerabilities.*"

Although these comments are probably correct, our results do seem to suggest that different frameworks affect the issues that subjects considered. While this effect may interact with experience, the choice of analysis framework sets the stage for not only the quality of the resulting analysis, but also its breadth and thoroughness.

Across the three experiments we saw a clear correlation between time-on task and the number of vulnerabilities discovered, across all frameworks tested. In the previous section some anecdotal evidence was presented for how STRAP performed in the case of high levels of expertise. How sustainable this trend is in the face of limited or no domain experience (either application domain or privacy), or in the case of other frameworks is unknown.

On one hand, subjects across conditions, as a group, were able to discover a large proportion of all known vulnerabilities in the target system. Table 21 shows how subjects in the different conditions performed on the different experiments, on average, and as a group. The gap between these two numbers shows that there is margin for improvement. This suggests that there is significant room for improvement, and that the average untrained subject does have the skills and the knowledge to identify most problems.

On the other hand, post-experiment interviews indicated that most subjects, across conditions halted their analysis when they did not because they had reached some pre-determined minimum time requirement, but because they felt they had explored all options for analysis, especially in the case of the Bellotti & Sellen and the Patrick & Kenny conditions. That subjects felt this way does not mean they really had exhausted all problems, but that they lacked support or incentives to continue beyond this point. The application of these frameworks, like most analysis methods, benefits from the kind of perspective that comes with time and repeat reviews of ones work. It is therefore possible

that these subjects, like the experts reviewing Augur, would have benefited from reviewing their analysis at a later date.

From the discussion of the results of the expert review in section 7.1 however, it becomes clear that the differences between experts and novices are deeper than simply more time on task. Experience, whether with the application domain or in privacy analysis, significantly affects the efficiency of analysis, with analysts discovering an average of 8.33 vulnerabilities per hour, whereas our subjects groups never managed to average more than 3.1 vulnerabilities per hour spent on their analysis. Therefore, while a team of inexperienced analysis may be as effective as a small group of experts, they will never be as efficient.

7.3 Frameworks and Quality of Analysis

The goal of this section is to summarize, for each of the frameworks tested, the relative strengths and weaknesses of the approach, where possible identifying what issues subjects struggled with, and why. We will compare the observations and predictions made in Chapter 3 with the results seen from the three experiments in Chapter 6, and point out new areas for study. I will also discuss the quality of the analysis subjects did with each of these frameworks, and the issues they helped identify and which they ignored.

The rest of this section is divided into sub-sections, starting with a general review, and then addressing each of the four frameworks tested in more detail.

7.3.1 General Review

As discussed in section 7.2, the overall quality of analysis was good given untrained subjects. Through the three experiments in chapter six, a strong case has been

made for the application of these frameworks as part of the overall design process, especially in domains where privacy can be an expensive or serious concern. Overall, the commitment and expense required from any of these frameworks, with the possible exception of the Patrick & Kenny framework which will be discussed in more detail later, proved to be fairly negligible (a 90 minute commitment). This being said, STRAP, derived from a meta-analysis of frameworks and the types of problems most users encounter proved to be the most successful framework, lead to the discovery of significantly more problems across all three experiments, at no additional cost.

Table 22: Experimental Results – Average Detection by Vulnerability Type

Average individual chance of discovering unique vulnerability, by framework and category.
Above average results bolded, results less than 50% of average highlighted in red and italicized

	Bellotti & Sellen	Patrick & Kenny	Risk Models	STRAP	Average (count)
Security	*12.19%*	**40.0%**	23.53%	**43.48%**	29.8% (13)
Notice/Awareness	21.36%	*15.0%*	**30.59%**	**35.58%**	25.63% (17)
Choice/Consent	**25.96%**	20.0%	19.13%	**24.92%**	22.5% (9)
Enforcement/Redress	**13.45%**		*4.55%*	**11.11%**	9.7% (4)
Weighted Average	18.81%	22.21%	23.64%	**33.46%**	24.76% (43)

The biggest problem with the combination of these frameworks and subjects was that there was a tendency for shallowness, for only considering the most superficial levels of the system, the most visible components. This was particularly apparent in the STRAP

conditions where we could actually see what functionality was considered, though we can infer that this was the case across conditions by the types of vulnerabilities discovered and the language used to describe these.

By grouping vulnerabilities in all three experiments together and classifying them according to the four FIPs categories, using the definitions given in (FTC, 2000), we can determine the relative strengths and weaknesses of the different frameworks at addressing different types of vulnerabilities.

Across all frameworks the detection-rate for all except enforcement-redress type vulnerabilities is in the 20-40% range, a detection rate which could be considered acceptable given the lack of experience and domain knowledge of our analysts, and when compared to the success-rates of Heuristic Evaluations (35% average detection rate according to Nielsen & Landauer (1993)). All four frameworks are weak at identifying and addressing enforcement/redress types of issues. This is also the category of problems which analysts, experts as well as novices, identified the least unique vulnerabilities. This may not be coincidental; it may simply be that analysts using these frameworks are prone to overlooking these types of issues. We also see that subjects using STRAP, while weak on enforcement/redress problems like subjects using other frameworks perform at above-average across the board.

Over the three experiments and four frameworks tested 85 subjects were recruited to perform an analysis of one of three systems, chosen to emphasize different properties of interactive systems. Subjects spent on average an hour and a half on their analysis, meaning that the whole subject pool donated over 127 hours of their time, or the equivalent of 16 full work-days.

7.3.2 Bellotti & Sellen

The Bellotti & Sellen (1993), and later the Bellotti (1997) frameworks are the oldest and most established points of reference. Despite their age and the speed with which computing evolves, subjects in this condition performed on par with subjects in the newer Hong et al. condition.

The Bellotti & Sellen framework is by far the simplest framework procedurally of the four tested in these experiments. It consists of a set of four key questions used to identify problem areas, and nineteen heuristics which an analyst applies to identify the problems and potential solutions. As such, this framework requires very little training or background knowledge to learn, though applying it successfully may require more training and insight given the limited support provided by the framework.

Because of this lack of structure and rigor it was hypothesized that subjects in this condition might have an advantage over subjects in conditions such as STRAP and Patrick & Kenny because of the overhead they require in structuring and analyzing the problem domain, especially when subjects spent little time on the analysis. This turned out not to be the case, even at relatively low levels of investment, with subjects in the STRAP condition outperforming those in the Bellotti & Sellen in both experiments where they were paired against each other. This was highly surprising; we had expected subjects in the STRAP condition to suffer from an initial ‘ramp-up’ cost due to the goal-oriented analysis (40% of the total time spent on their analysis). This allowed subjects in the Bellotti & Sellen condition to spend more time on the task of identifying vulnerabilities. However, subjects in the STRAP condition seem to reap the benefits of this investment surprisingly quickly, even overtaking their Bellotti & Sellen peers in a 90 minute trial.

While the Bellotti & Sellen subjects had more time to spend on their analysis, this time was spent less efficiently.

From reviewing the types of vulnerabilities uncovered using this framework, we see that subjects using this framework did best at detecting notice/awareness and choice/consent issues, though in the first category subjects performed below average when looking across all categories. Subjects using this framework did exceptionally poorly at identifying security vulnerabilities compared to their peers. The framework does not emphasize this point, nor does it help the designer consider the mechanics of the system to be built, so this result was as expected.

One interesting thing to note is that though this is by far the oldest and most basic framework, it performs roughly in line with the Hong et al. and Patrick & Kenny framework in terms of overall efficiency. This is especially surprising given that this framework was taken out of the context of ubiquitous computing systems for which it was proposed. This seems to imply that though the authors do not make claim to this effect, they did a good job at selecting heuristics which would be widely applicable.

7.3.3 Hong et al.'s Risk Models

The framework with the closest resemblance to the Bellotti & Sellen framework (1993) is the 'Risk Model' framework developed presented by Hong et al. (2004) This framework, while somewhat more structured is still based around the notion of a checklist of questions and issues which the analyst should consider, and which will help identify potential problems and strategies for their resolution. Like the Bellotti & Sellen framework it also builds on the study of ubiquitous computing systems, though the definition is less strict in this case.

This framework was by far the most popular one with subjects in the third experiment, as evident from the questionnaires, with subjects rating this framework highest in terms of ease of learning, use, and perceived utility (see section 6.3). Part of the popularity comes from the fact that Risk Models gives the user a more defined procedure to follow by giving them more detailed questions to consider. This means that analysts have to rely less on their own instincts or insight, though they are free to stray from the path at any time. For the novice analyst, this added structure is a definitive advantage.

One of the problems identified in chapter three, and pointed out by subjects in the interviews and post-experiment questionnaires was the cost-benefit analysis step of the framework. Kenneth, one of the subjects in this condition explained: "*I kinda think that it* [the cost benefit analysis] *came out of left-field, I really do. It seemed like such a small portion of the analysis. It seemed like it was trying to add something that would make some sort of decision in the paper, but I don't think the rest of the paper is really geared towards that."* Kenneth objected to two things in particular about this step: the first was the imprecision inherent in this step, and the second was that it was often difficult to estimate the risks before going to what Hong et al. defined as the next step in the analysis. The next step asks analysts to consider how unwanted disclosures are taking place, what the default settings of the system are in terms of information sharing and use, and what the users' priorities or sensitivities will be in this situation. Kenneth's is a very interesting and valid complaint. There seems to be some overlap between the identification questions and the questions meant to identify design solutions.

Gregorio, the second subject interviewed in this condition agreed with this analysis: "*Well, I think, first time I read it I found it pretty strange. Considering that for*

example estimate of damage is very precise. I think it does make sense that, that it is pretty logical that the likelihood, but how well can it be presented mathematically [accurately*], how do you estimate these likelihoods ahead of tim*e*?*" Gregorio went on to reflect that there were deeper, more fundamental flaws with this type of analysis.

We started by discussing the problem represented by tipping points and near-equilibrium problems (vulnerabilities which have roughly equal costs and benefits associated with them), and what a designer should do in these unclear situations. From this starting point, Gregorio started to talk about interactions. He realized that some types of vulnerabilities and information disclosures may be relatively harmless in isolation, but that when occurring together the damages may exceed the sum of the two in isolation. For instance, letting someone know your zip code may be relatively harmless. The same could be said for letting someone know your birth date or your last name. When taken together, these two pieces of information can almost certainly enable a determined third party to identify who you are, including your home address and phone number. The conditional risk, the risk associated with the surname leaking given that the zip code has been disclosed, is higher than the sum of the two individual risks, something which cannot be modeled in this simple cost-benefit analysis. It is also true that the chance that one's surname will fall into the wrong hands given that one's zip code has been misappropriated is often a conditional risk greater than the sum of the two individual events.

Hong et al. emphasized that the numbers in this formula need not be accurate: *"[…] the utility of this cost-benefit analysis comes not so much from accurate and precise values, but from having the design team think through the issues of likelihood, damage, and cost […]"* (Hong et al., 2004). Nevertheless, these kinds of flaws can

seriously undermine the value of including this step. This objection has wider implications than for Hong et al.'s framework, as even the simpler cost-benefit analysis advocated in STRAP may be affected by the same issues. This possibility will have to be examined more deeply in later investigations.

Despite subjects' enthusiasm for the framework, they also expressed doubts as to exactly how valuable or objective a form of analysis it really was. As Kenneth put it: "*I think it* [the results of the analysis] will *depend on who you ask, I think you'll get a wide spectrum of issues and results, I think you bring your own perspective, and I don't think these questions change that.* […] *If the goal of the analysis is for me, at a company, to analyze the entire system for privacy problems, then I think it's* [bringing in your own perspective to this extent] *a real detriment, because then it's my view of the system. How many people then do you need to have analyze the system to ensure good coverage?*"

Looking at the analysis that people did across conditions and the variability in the vulnerabilities they reported, this is a characterization that it would be fare to make about most of the frameworks described here. When we look at tables 15, 18 and 20 and see the number of vulnerabilities with very low chance of detection, it becomes apparent that individual analysts, their perspectives and their experiences do play a significant role in what the analysis revealed. Design and analysis are both creative processes, and therefore depend on the creativity and spark of the analyst-designer.

Despite continuously describing the analysis part of the framework (discovering problems) as an objective process, Gregorio admitted that "*different people will bring very different analytical perspectives to the table, and their analysis will not match, which is a problem.*" Gregorio also admitted to having performed a little experiment of his own. Before reading the framework description he had simply read the system

description and for the next few minutes tried to think and write down as many privacy problems associated with the design as he could think of. Gregorio then went on to read the description of Hong et al.'s framework and applied this to the design. He admitted not having found any significant new problems compared to his freehand list, though the framework did help him refine what it was that he had originally written down, and define exactly when and how these issues would become problems. It should be noted that Gregorio was a very bright and successful student, and performed well in this experiment, and continued to enthusiastically advocate the use of Hong et al.'s framework despite the fact that it had not helped him personally.

Table 22 shows that subjects using the Hong et al. framework generally did well across the board, except in detecting enforcement/redress issues. While all frameworks performed badly on this front, Hong et al. was the worst. The Hong et al. framework seemed generally unaffected by the choice of problem domain in the experiments, and was general-purpose and adaptable enough for subjects.

7.3.4 Patrick & Kenny

The Patrick & Kenny framework (2003) is the only framework tested in only one experiment, and being the framework for which we therefore have the fewest observations. In part this is an artifact of the speed of development in this field. The Patrick & Kenny and Hong et al. frameworks were not published until after the start of this research. Because of this, the first experiment in the series only featured two frameworks, STRAP and the Bellotti & Sellen frameworks, a reasonable selection given the frameworks available at the time (this research started late 2001, early 2002). The last

two experiments naturally had to be re-designed to allow us to compare more frameworks.

This being said, and as evident in the analysis in Chapter 3, the Patrick & Kenny framework, because of its combination of heuristics and structured analysis led us to have high expectations to its performance. Of the five frameworks examined in Chapter 3, the Patrick and Kenny framework was that which most closely resembled the final version of STRAP. While many of the theoretical assumptions and decisions that went into these two frameworks were similar, the results of their application proved to be dramatically different.

Of the frameworks tested, the Patrick & Kenny framework received the lowest ratings in terms of clarity, ease of use, ease of learning, and general usefulness. Part of the reason for this was that subjects were generally unable to get beyond the focus on understanding the legal requirements of the domain, and the user interface design. Both the interview subjects in this condition agreed on this point. One of the subjects, George, explained what he thought of the framework and why it was a poor match to the Teamspace system: "*It depends on the environments, I guess if there was legislation that says these privacy requirements have to be met and in that case this particular procedure works, I mean it tells you what you need to* [do]." These echo Matt's sentiments, discussed in chapter 6.3.

Another problem associated with this method was that the overhead associated with the use cases and object-sequence diagrams was too high. None of the five subjects in this experimental condition derived more than a single use-case and object-sequence diagram. The one use case and diagram subjects derived served more as a demonstration of the fact that they knew how to do the analysis rather than serve as the foundation for

the analysis. The general consensus, as discussed in section 6.3 was that it was too labor-intensive a task to perform at this stage of design. If such an analysis had been derived as part of some other analysis, the method might have been more appealing.

This is really an important point. Deriving only a single object-sequence diagram, subjects spent the same amount of time on structuring the problem as subjects in the STRAP condition who mapped out most of the systems functionality, 40% of overall time on task, on average 55 minutes (see Table 19). This is a significant portion of the overall time on task, and deriving more object-sequence diagrams was likely prohibitively expensive to subjects.

This is related to the problem identified with this framework in Chapter 3, that there are no guidelines or procedure for identifying vulnerabilities from the object-sequence diagrams. This naturally lowers the appeal of actually deriving these, and subjects may have decided that they were better off simply applying the heuristics on the use cases given in the system description. Given the amount of time subjects invested in deriving a single use-case and object-sequence diagram and the question of their overall value in the analysis, subjects were justified in questioning the purpose of performing this analysis and skipping to simply applying heuristics to an unstructured problem-space.

One potential confound in all this is the way the system was described, and by this I mean the form of presentation rather than the information conveyed. As seen in Appendix C, the Teamspace system was described through little scenarios or use cases. While this should have made it very simple for subjects in this condition to identify and document their own use-cases, it may also have made it less necessary for them to do so. Subjects may have felt that the description provided was adequate enough for them to work on directly rather than trying to structure it further.

Table 22 shows that though the framework overall performs on par with the others and is strong on the identification of security issues, it is surprisingly weak on the identification of notice/awareness types of problems. This is surprising given the emphasis this framework puts on understanding the usability requirements of privacy, of which notice and awareness is an essential part. The post-experiment interviews did not reveal any reasons for this curious effect.

Overall the performance of the subjects in this experimental condition was disappointing given its similarity to STRAP, but the low number of subjects and single experiment means more research is needed before any firm conclusions can be reached.

7.3.5 STRAP

Of the frameworks presented in this thesis, STRAP stands out for a number of reasons. The creation of the STRAP framework is a central part of the thesis. STRAP serves both to cement and test theories about why different frameworks work and fail by putting our knowledge and understanding to the test in actual experiments with existing frameworks. As such, STRAP closely resembles parts of other frameworks discussed in this thesis, as well as borrowing form other methods and techniques. The approach taken in the development of STRAP was definitely more evolutionary than revolutionary, sticking to elements which we knew worked or had the properties we found desirable and integrating them together.

One of the things that make STRAP stand out from other theories is its use of a goal-oriented analysis as the back-bone for the analysis. While goal-oriented analysis is accepted and used in requirements engineering, it is not a method we have only exceptionally seen used in this field (notable exceptions being hierarchical task analysis

and GOMS methods). This particular form of goal-oriented analysis is therefore not a technique with which many analysts will be familiar. While we knew that goal-oriented analysis could help us overcome the problems we saw with design fixation by helping analysts map and understand the overall goals and functionality the system needed to support, we did not know whether it would be a technique which analysts could master, or would embrace.

One of the critiques we had of the *i** framework (Yu & Cysneiros, 2002), and one of the likely reasons why the Patrick & Kenny framework (2003) failed in these tests was that though the techniques they advocate have been proven in the requirements community, they are considered too onerous for this type of analysis, especially when compared to frameworks like Bellotti & Sellen (1993) and the Risk Models (Hong et al., 2004). While exactly where to set this threshold depends on who the analysts are, their motivation, and training, we considered it crucial to ensure that analysts would not be turned away from the method because of this step.

Goal-oriented analysis and the structure of STRAP was embraced surprisingly well in these experiments. As one of the subjects in experiment three, Stacy, said: "*That* [goal decomposition] *was really easy, that wasn't hard at all, I was able to like draw up all the goals really easily.*" Stacy was at this point two years into her career as a student in computing, and did an excellent job in her analysis. This sentiment was echoed by other subjects, and reflected in the experiment survey (see section 6.3) where it scored highly on a number of desirable attributes. We also saw from the analysis of experiment one (section 6.1) that subjects quickly and competently employed this analysis method (see figures 37 and 38).

Subjects in the STRAP condition consistently outperformed those in the other conditions in all three experiments, giving them both a higher chance of discovery in most categories of problems, but also a wider range of the types of vulnerabilities discovered. While this is an important result both validating our design decisions and lending arguments for the promotion and use of this framework, it is important that we go the extra step and ask why this framework proved so effective, and what improvements can be made to it.

In addition to not providing a burden to the analysts, the goal-oriented analysis seemed to help them in their task. As Stacy put it: "*The Tree helped, I'm a very visual person, and whenever I'm doing anything it helps if I draw things out and see how it works just visually, and there I think the tree helps.* […] I*f you really do it right, you can definitely list all your goals and all your actions and expose things that way, I think it would be a really effective method.*" Stacy described how she went through the system description, and from each scenario identified goals based on the interactions described. This is rather like the approach advocated in GBRAM (Antón 1996, Antón & Potts, 1998) and ScenIC (Potts, 1999) which this goal-oriented analysis method is based on. While we do not specifically require the use of scenarios in STRAP, fearing it would prove too much of a burden, they are a great source for understanding and identifying goals for this analysis. Therefore, when available, they should be taken advantage of. What is interesting is that scenarios alone (as in the Patrick & Kenny condition) do not seem to spark the same cognitive processes.

Experiment 2 provided an interesting contrast to the loyalty subjects showed this method in the other two experiments in that a small group of subjects decided to do the analysis without performing the goal-oriented analysis. Instead, this group of students

simply performed a very minimal hierarchical task-analysis (see figure 42, and section 6.2). This is an even simpler breakdown of the system than goal-oriented analysis, and subjects defended their actions by saying that they were so familiar with the application domain that this was all they needed to guide them through the analysis, serving as a checklist of the high-level functionality they needed to consider. Again, this is an indication that the goal-oriented analysis does require a price in terms of time and effort, and that there is a threshold at which analysts are no longer willing to pay this price. This experience serves to demonstrate that the perceived complexity of the system is one of the variables which control these decisions.

The opposite argument is that goal-oriented analysis is a less formal method and notation than use-cases and object sequence diagrams as used on the Patrick & Kenny framework (2003). It could be argued that using these more formal methods would provide a better foundation for analysis than a goal-oriented analysis. The evidence from Experiment 3 shows this is not the case. Experiment 3 also shows that subjects in the Patrick & Kenny condition spent an inordinate amount of their time, the same as invested in the goal-oriented analysis of the whole system, deriving a single use-case and object-sequence diagram. This highlights the gulf between the type of investment more formal methods such as Patrick & Kenny require compared to STRAP.

However, basing ones' analysis off the results of more formal methods might be a more compelling argument for a design and development team using these more formal techniques to identify and analyze the rest of the system behavior, meaning that this analysis does not need to be performed solely only to identify privacy vulnerabilities. Conversely, the benefits of STRAP would likewise be enhanced on projects using goal-based tools for requirements engineering and task analysis.

While this is an issue that would definitely require more research, it is not entirely clear that using a more detailed description of the system would lead to a better privacy analysis in either case. When looking at privacy vulnerabilities, many components interact with each other to determine whether there is a risk, and how high that risk is. As such it is necessary for the analyst to have a global picture of what is going on. While this is just conjecture, it is possible that an analyst basing his or her analysis on too detailed a mapping of the systems functionality and requirements would be unable to see the forest for the trees. Again, this is pure speculation, and something which will need to be examined in more detail in the future.

One of the interesting questions raised early in this research was how detailed a goal-oriented analysis subjects would perform (see section 6.1 for a discussion), and whether this level of detail would not only affect the number, but also the types of vulnerabilities discovered. In other words, would it be the case that as goals are broken down into more detailed sub-goals, architectural and technical details are exposed and decisions are made, thereby allowing the analyst to discover and document not only more problems, but a different class of problems.

This debate is by no means new, and analysis frameworks breaking tasks and operations down into different levels detail are available, and are used for different purposes in the design process. One examples of this is the level of detail and the types of problems identified in mainstream GOMS versions (John & Kieras, 1996) vs. the Keystroke Level Model (KLM) (Card et al. 1980). Whereas the first focuses on the more high-level operational units users have to perform (much like in the goal-oriented analysis used in STRAP), KLM looks at user actions on the level of individual key-

strokes and operations. Both methods are useful and provide insight into the problems users might face, but at a very different level.

For instance, while the KLM might help an analyst examine the efficiency of a set of operational sequences, it would likely be less useful for looking at whether the goals and cognitive models users have are being supported. Likewise, while the GOMS analysis will likely help an analyst break down how user goals are to be met, and how they meet users' needs, including how these operations match their cognitive and perceptual processes. This is something which an analyst using the KLM-level of analysis would not be able to investigate.

Another way of thinking about this is to consider what happens in the border area between analysis and design, and whether the types of problems discovered changes. As goals are broken down into ever more detailed sub-goals, it is possible, desirable, and necessary, to make certain architectural decisions in order to continue this decomposition (for instance deciding whether the system should have a central server, a central database, or whether the system will be distributed). Once this level has been reached, it is sometimes possible or desirable to go deeper, to talk about the use of specific technologies or functions (email encrypted using PGP, or data sent securely over SSL, or a data stored in a specific database system such as MySQL). This in turn could help identify new vulnerabilities associated with this way of operating, or with the technologies chosen. Somewhere along this process we have gone from analyzing the problem space and system to designing the system. The question then becomes, do you need to reach the level of specifying technology goals in order to identify technology vulnerabilities?

While the three experiments were not specifically designed to answer this question, an attempt was made to mine the data and diagrams from the three experiments to help answer this question. All the goals in all the goal-trees of the three experiments were put into one of three categories; abstract, architectural, or implementation-oriented goals. Abstract goals were defined as goals which primarily identified some agents' goals, intent or need (for instance, system determines shortest route between origin and destination); architectural were goals which expressed the need for certain system actions to take place (for instance, system queries database for road information); and implementation-oriented goals were defined as those goals specifying a set course of action or implementation (for instance in Browsing, route information is encoded in HTML and sent to user over HTTP).

Vulnerabilities were similarly classified according to the types of vulnerabilities they express, whether abstract (for instance from Browsing, the lack of awareness of information practices), architectural (for instance in Browsing, privacy policy placing too high a burden on users, or DNS request allows 3rd party monitoring), or technical (for instance from Browsing, communication not being encrypted, or local cache security). The goal was then to see whether there is a correlation between the types of goals and the types of vulnerabilities discovered.

Given that the data was not collected, presented, or generated with this purpose in mind, the mappings proved to be somewhat tricky, though a fair and honest effort was made to classify all goals and vulnerabilities. The 34 subjects in the STRAP condition returned 27 goal-trees (10 from experiment one, 11 from experiment two, and 6 from experiment three). Of these, six of the diagrams from experiment two were excluded because they only consisted of four goals and could hardly be called goal-trees. A total of

21 diagrams were therefore analyzed, containing a total of 252 goals and 117 vulnerabilities. This is an average of 12 goals per diagram (the five remaining diagrams from Experiment 2 still dragging down the average), and an average of 5.57 vulnerabilities per diagram, which is about 10% more than the average across all STRAP subjects.

In cases where goals or vulnerabilities could not unambiguously be classified as belonging to a single category, they were excluded from the analysis. This only occurred with two vulnerabilities (1.7%). Table 23 shows a breakdown of goals and vulnerabilities in each category, and table 24 gives a breakdown of the correlations between the frequencies of occurrence of these categories.

Table 23: Goals and Vulnerabilities by Detail Category

		Overall	Abstract	Architectural	Technical
Goal	**Sum**	252	186	59	7
	Average	12.00	8.86	2.81	0.33
Vulnerability	**Sum**	117	54	32	24
	Average	5.57	2.57	1.52	1.14

In table 24 we see that there are no strong correlations between the types of goals expressed and the types of vulnerabilities associated with them. The only correlations are weak, between the number of architectural vulnerabilities on one hand, and the total number of goals or the number of abstract goals on the other. While this may seem counter-intuitive, these findings are plausible. One must keep in mind that the top levels

of decompositions tend to be abstract, followed by architectural goals and then technical goals. The more goals therefore, the more architectural details can be addressed.

Table 24: Correlation between Types of Goals and Types of Vulnerabilities

Weak correlations bolded

		Goals			
		All	**Abstract**	**Arch**	**Tech**
Vulnerabilities	**All**	0.423	0.371	0.309	0.123
	Abstract	0.267	0.260	0.162	-0.097
	Arch	**0.553**	**0.512**	0.366	-0.029
	Tech	-0.248	-0.351	0.207	-0.142

The reason why we don't see any other correlations is a little more difficult to explain. First of all, one should keep in mind that only 2.7% of the goals were classified as technology oriented. This is simply too small a sample to determine any form of correlation. Abstract vulnerabilities on the other hand are less bound to any category of goals, as any type of goals can cause abstract vulnerabilities. Many abstract vulnerabilities occur in tandem other, more technical vulnerabilities, referring to the users' potential lack of awareness.

It is also interesting to note that many of these analysts tended to short-circuit their analysis, possibly the reason why so few technical goals were documented. A typical example is then subjects identified a need to communicate with a database or a server (an architectural goal). In these cases, they typically identify and attach the technical vulnerabilities associated with the technical implementation directly on the architectural goal. In other words, rather than breaking things down to the lowest level, subjects would end the decomposition as soon as they understood the requirements without necessarily writing these down.

From this data we must conclude that there is either no threshold or level of analysis subjects must reach in order to detect more technical problems, or at least that such a relationship is more complex than the classification used in this analysis. Another possibility is that too many of the subjects in this sample did not take their goal-decompositions seriously enough, and that such a relationship may emerge in the hands of more experienced analysts.

CHAPTER 8
CONCLUSIONS

The goal of this project was to gain understanding of how people think about and manage their online privacy as well as to provide the right tools to designers in order to address these privacy issues. In this dissertation I have presented a number of analyses and studies of privacy management practices and interfaces, of privacy design frameworks and techniques, and the mismatches that occur. As part of this analysis I defined a new analysis framework, STRAP, integrating the lessons from these studies. The goal behind developing STRAP was both to test the assumptions and conclusions of these studies as well as advance the state of the art in privacy-aware design by providing analysts and designers with better tools As a consequence, the research statement was centered on showing that STRAP performed better than existing methods, specifically the Bellotti & Sellen (1993), Hong et al. (2004), and Patrick and Kenny (2003) frameworks, in terms of being:

1. Applicable to a broader class of systems
2. Accessible to analysts with little experience in the method or privacy analysis
3. Efficient in terms of the time-on-task needed to produce acceptable results

As was demonstrated in chapters six and seven, STRAP does help analysts do a more thorough analysis than these other frameworks, at least for the problem domains examined, and with inexperienced analysts. The surprising thing about these experiments was that all frameworks seemed to hold up well across domains, even those derived from the study of very specific and specialized systems. It is therefore hard to say whether STRAP does any better or worse over a variety of domains, as the frameworks it was

compared held up well as well. What we can say instead is that the selection of frameworks examined were less domain specific than originally thought.

Apart from the validation and testing of STRAP, this work was meant to demonstrate that cost-effective and efficient privacy-aware design and analysis frameworks are available. In general, subjects in the STRAP trials discovered, on average, one third of known privacy issues (33.46%), which is higher than the success rate for the other frameworks. Objectively speaking however, this is not a high success rate, especially when compared to what would be considered acceptable for analyzing mission-critical or safety applications for flaws or problems with more formal methods.

What does speak in favor of these frameworks, and especially STRAP is that relatively small teams of untrained analysts are able to combine their analysis for a much higher success rate. As was the case in heuristic evaluation (Nielsen & Landauer, 1993), while the individual analysts' performance might be relatively poor, groups of analysts could provide acceptable results. In the case of STRAP, teams of 4 analysts spending on average 90 minutes on their analysis would on average identify between 70% and 80% of all known vulnerabilities. While there is no data available on the amount of time each subject spent in Nielsen & Landauer's experiments, their individual and group performance is on par with those seen for STRAP (35% individually, 73% for teams of 4 analysts).By the standards set by Heuristic Evaluation, an accepted usability method considered efficient and effective, STRAP should be considered efficient and effective as well. While Heuristic Evaluation is primarily used in the evaluation of interfaces, the costs and risks associated with user error can often be as high as for privacy problems.

While STRAP led to better results, any of the frameworks examined here should prove a worthwhile exercise for designers. The amount of time and effort required to get

useful results is usually minimal, and the potential benefits can be significant. The three experiments carried out as part of this thesis work should serve as proof of the fact that these methods work, and to justify the investment in time and manpower needed to perform this type of analysis. As Gregorio demonstrated, just setting off to think about privacy issues early on in the design phase can be a fruitful exercise, even without the support of any framework.

While many questions remain to be answered about exactly why some frameworks succeed or fail, I believe I have demonstrated how the application of heuristics can be supported and even improved through the use of structuring mechanisms, in this case a goal-oriented analysis. I hope that this work will allow us to re-examine these very popular techniques, including an analysis of their relative strengths and weaknesses, and whether these can be addressed through the introduction of structuring techniques as in STRAP.

CHAPTER 9
FUTURE WORK

The goal of this thesis was to test a set of theories and assumptions regarding how people think about privacy, and how designers, especially novices, go about the task of designing for privacy. As part of this goal, the most important design frameworks in this field were analyzed, and the lessons learned incorporated into a new framework called STRAP, which was shown to be more effective, thereby validating the analysis. While an important point to prove, in the grand scheme of things it is also the most basic thing to prove, and the more difficult question is to demonstrate exactly why this works, or which elements of the design framework contribute to an improvement, and which do not. This is especially true for the methods used to structure the problem space.

In STRAP, analysts are asked to perform a goal-oriented analysis of "the system" in the big sense, including computers and software, users, third parties and outside elements. Goal-oriented analysis is a popular method in requirements engineering because it allows analysts to think and reason about the system at a very high and abstract level, without getting bogged down in the implementation details. Goal-oriented analysis, as prescribed in STRAP, was also easily adopted and used by subjects, generally leading to a high quality of analysis with little added effort or time on task.

The experiences from experiment two, where subjects were generally very familiar with the type of system they were asked to analyze, and their subsequent skipping or minimization of the goal-oriented part of the analysis begs the question whether such an analysis is always required. A small number of subjects decided instead to perform a more light-weight hierarchical task analysis, with no apparent loss of

performance, though it should be noted that the quality of goal-trees in this experiment in general was poor. The question therefore remains, is it always necessary to go to such lengths to structure the problem, should analysts be encouraged to make their own decisions on this, or should we instead outright advocate a different, more light-weight analysis method? How does the complexity of the system or the analysts' familiarity with it affect the answer to this question?

In the future, given time and resources, it would be interesting and valuable to investigate whether individual elements of the framework, if taken out of context, eliminated or replaced with other, similar methods, leads to an increase in performance. For instance, would replacing the goal-oriented analysis with a hierarchical task analysis lead to better or worse results? Would replacing STRAP's heuristics or putting them into Bellotti & Sellen of the Patrick & Kenny frameworks improve their overall performance? While somewhat tedious, it would be necessary work in order to determine how optimal an approach this is. This would also allow us to examine whether it is possible to somehow improve the detection-rate of enforcement/redress type issues, which were generally ignored in all frameworks.

It would also be interesting and valuable to investigate how these frameworks, and especially STRAP helps more experienced, and preferably more highly motivated designers and analysts go about their tasks in a more naturalistic setting. This would require getting people to use STRAP or one of these other methods in the design of real systems, and observing them in this process. Observational studies were something which were not pursued in this thesis work because of time and logistic constraints, but something which is definitely worth pursuing, especially in the light of the insight the

post-experiment interviews in experiment three provided into why a method works or fails, and which processes and tasks users struggle with.

Another reason for switching strategies in the manner described above is the lingering question over how big a role experience has on success, and what kind of experience comes into play. To get more experienced and motivated subjects, one typically has to go to them, observe their work practices and processes, basically watching them in action. This issue of the value and effect of domain/application experience vs. experience with these frameworks or privacy-related problems is especially interesting, and should be a fruitful and interesting avenue or research, as evident from the expert analysis of Augur. If it is confirmed that both types of experience are needed this could have important implications for the composition of design teams and the role of the organizations privacy officers. It could also affect how we design analysis frameworks, and what processes we try to scaffold in what way.

Should it be the case that application experience alone is insufficient to ensure a thorough analysis, it may be necessary or worthwhile to investigate whether these frameworks can be modified to compensate for a lack of experience with privacy issues. It could be the case that a different set of heuristics or analysis questions, perhaps based on the types of problems these analysts tend to miss, would lead to improved results.

Another interesting avenue of research would be to re-examine the concept of the cost-benefit analysis in light of the data that came to light in this study. Clearly, some form of decision needs to be made at some point about which solutions are worth pursuing and which are not, about which problems are worth addressing or should be given priority, and which should not. The question is whether analysts are best served with a quick and simple approach as advocated today, or whether it is worth developing a

model which would account for the types of conditional probabilities and compound effects we see in this problem domain. Many organizations and projects are faced with strict budget limitations. For these it is essential to either restrict the amount of effort spent addressing privacy issues, or know where their investment will have the highest payoff for them and their clients. .

It is important to acknowledge the limitations of the research presented in this thesis, which centered on these frameworks abilities to identify privacy problems. Most of these frameworks also provide mechanisms for identifying design solutions, something which was not evaluated experimentally as part of this work because design typically takes longer than analysis, and would have required a significantly higher investment on the part of the subjects. It would also have presented challenges in terms of how to objectively and thoroughly evaluate these design solutions. While challenging, this may be work worth looking into to get a better sense of the real value of these different approaches.

In this dissertation I also presented three studies into how people think about privacy, what problems they encounter, and how current user interfaces support or fail to support the processes and processes the users need to perform in order to manage and protect their privacy. This line of research should be pursued, taking the lessons from the existing studies and looking at how these lessons can be applied to the design of awareness and management interfaces, especially in conjunction with the work on iWatch.

Currently, iWatch monitors a total of 21 common privacy practices websites engage in. Users are rather simply informed of the sites practices through one of two toolbars, which present some graphical or textual indicator for the user to see. The reason

for developing two interfaces was specifically to test and see what kinds of tradeoffs users make in terms of the level of detail of the information they want, and the cost, the amount of distraction and screen real-estate they must sacrifice to obtain it.

More advanced awareness interfaces, including those with more intelligent rules about when to interrupt or involve the user are technically feasible. Such interfaces would require either a learning system to study the users actions or an advanced preference management system allowing users to effectively specify their own privacy policies. Such a system presents important challenges in terms of design as it needs to be highly flexible, while at the same time be accessible to people who are neither privacy experts, nor programmers. Most of the systems allowing for the kind of flexibility needed in iWatch requires the user to script their own actions in a language like Perl, something which is clearly outside of what can and should be expected of the average computer user. Coming up with new and innovative management metaphors and models is a challenging, but ultimately worthwhile pursuit.

APPENDIX: OVERVIEW

The following appendixes contain the descriptions of the systems given to subjects in Experiment 1 (Appendix A) and 3 (Appendix C). These descriptions were written by third parties, and are included for reference with due credit and permission. The description given to subjects in Experiment 2 is given in section 6.2.1. Appendix B contains a copy of the questionnaire given to subjects in Experiment 3, again for reference purposes.

APPENDIX A: AUGUR

By Joe Tullio

Introduction

Augur is a shared calendar system designed to help a group of colleagues communicate. Calendars are often used as tools for assessing someone's availability or location, but they require maintenance to remain accurate. For instance, a person may not attend all the events he schedules, or he may schedule events that conflict one another. A student may have stopped attending a particular class or seminar even though those events remain on her calendar. Problems like these make a calendar less useful for the communication tasks it typically supports, such as finding particular colleagues or scheduling time with them.

Augur is a web-based, shared calendar that provides additional predictive features intended to facilitate communication within a workgroup. These features include predictions on the attendance of colleagues at future events, as well as predictions on who has scheduled the same events. These predictions improve over time by learning from past attendance patterns. With these features, users can identify events that are no longer attended, make informed decisions about which of several conflicting events will be attended, and determine who they will likely see at a particular event.

Interface

Users access Augur by opening the Augur URL in their browser and securely logging in. To ease the login process, Augur is capable of automatically logging in users from a particular computer. Once logged in, users are presented with a welcome screen allowing them to navigate to their calendar for particular day.

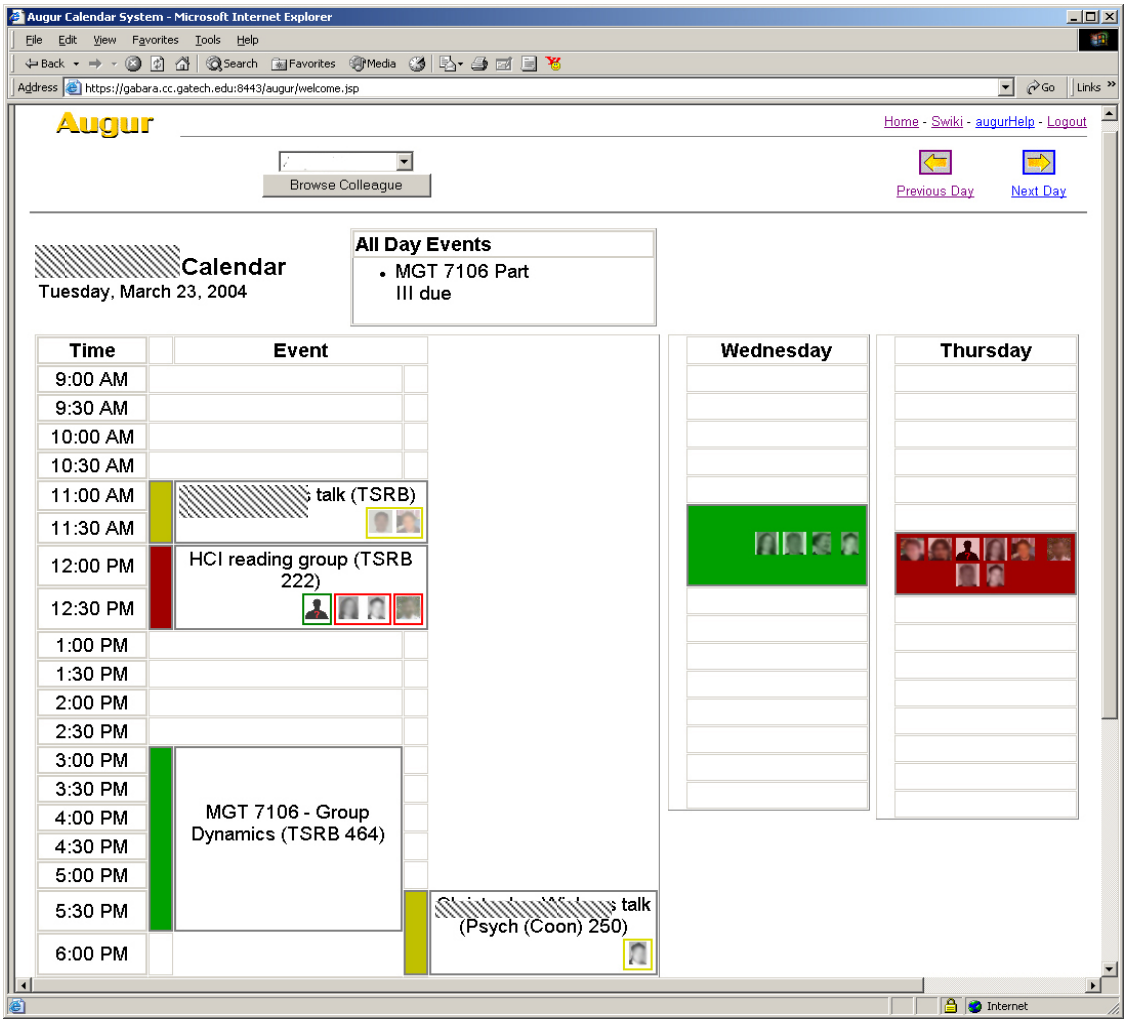


Figure 45: Augur Interface Example 1

Augur presents a user's scheduled events for a day in an hour-by-hour, block format that is similar to the tabular style used by other calendar systems (iCal, Outlook, Mozilla Thunderbird, etc.). However, this view is augmented with additional information that indicates colleagues who have scheduled the same events and attendance probabilities for colleagues at those events. Figure 45 shows a screenshot of Augur.

The events on a user's calendar are augmented with a list of icons that indicate which of the user's colleagues have also scheduled the event. Each icon represents a particular colleague, and a colleague's icon is displayed within an event on the calendar if the colleague has also scheduled that event. In the calendar shown in Figure 45, the user can see that four other colleagues have co-scheduled the 'HCI reading group' event on their calendars. Icons are arranged in decreasing likelihood of attendance from left-to-right. Colleague icons are clustered in an event based on their attendance likelihood using colored boxes; the color of the box around an icon group indicates the attendance likelihood of the colleagues in that box. For example, a bright green box surrounds colleagues' icons that are very likely to attend the event. The color groups are bright green, green, yellow, red, bright red in descending order of attendance likelihood.

Events on a user's calendar also have a colored bar to their left that indicates the user's likelihood of attendance at that event based on Augur's predictive models. The color scheme used for this bar is identical to that used for the colleague icons described earlier.

To the right of the daily calendar are visualizations of the worker's calendar for the next two days, which we call 'bar calendars'. Note that the bar calendar does not display the events' descriptions. Event blocks in the bar calendars are colored to indicate the overall popularity of an event; again, a green, yellow, and red color palette is used to

color the bar calendar's event blocks. An event's popularity is sum of the attendance probabilities of all colleagues who have scheduled the event. Hence, events where the worker is likely to see many colleagues are colored green, events where the worker is likely to see a few colleagues are colored yellow, and events where the worker is unlikely to see any colleagues are colored red. As in the daily calendar, we place icons in bar calendar event blocks to indicate which colleagues also have scheduled events that are on the user's schedule. Again, left-to-right ordering is used to indicate the likelihood that a colleague will attend an event.

The user can also interact with the calendar to obtain more information about his colleagues' calendars. When the user mousses over an icon on his daily calendar, a menu pops up. This menu identifies the colleague using his name and a small picture, indicates how likely the colleague is to attend the event, and provides a hyperlink to the colleague's calendar. When the user clicks on the hyperlink, an animation shrinks the user's calendar, hides the user's bar calendars, and displays the colleague's calendar to the right of the user's daily calendar (Figure 46). This allows the user to easily compare schedules and plan communication with the colleague accordingly. Note that the colored bars to the left of events on the colleague’s calendar are predictions of attendance for that colleague.

Augur system description

The Augur system consists of a number of components that process, store, and serve calendar information located in a central relational database. It retrieves user calendar data from other calendar systems such as Palm Desktop and iCal, augments the data with information about attendance likelihood and events co-scheduled by colleagues, and serves that information to the web-based user interface.

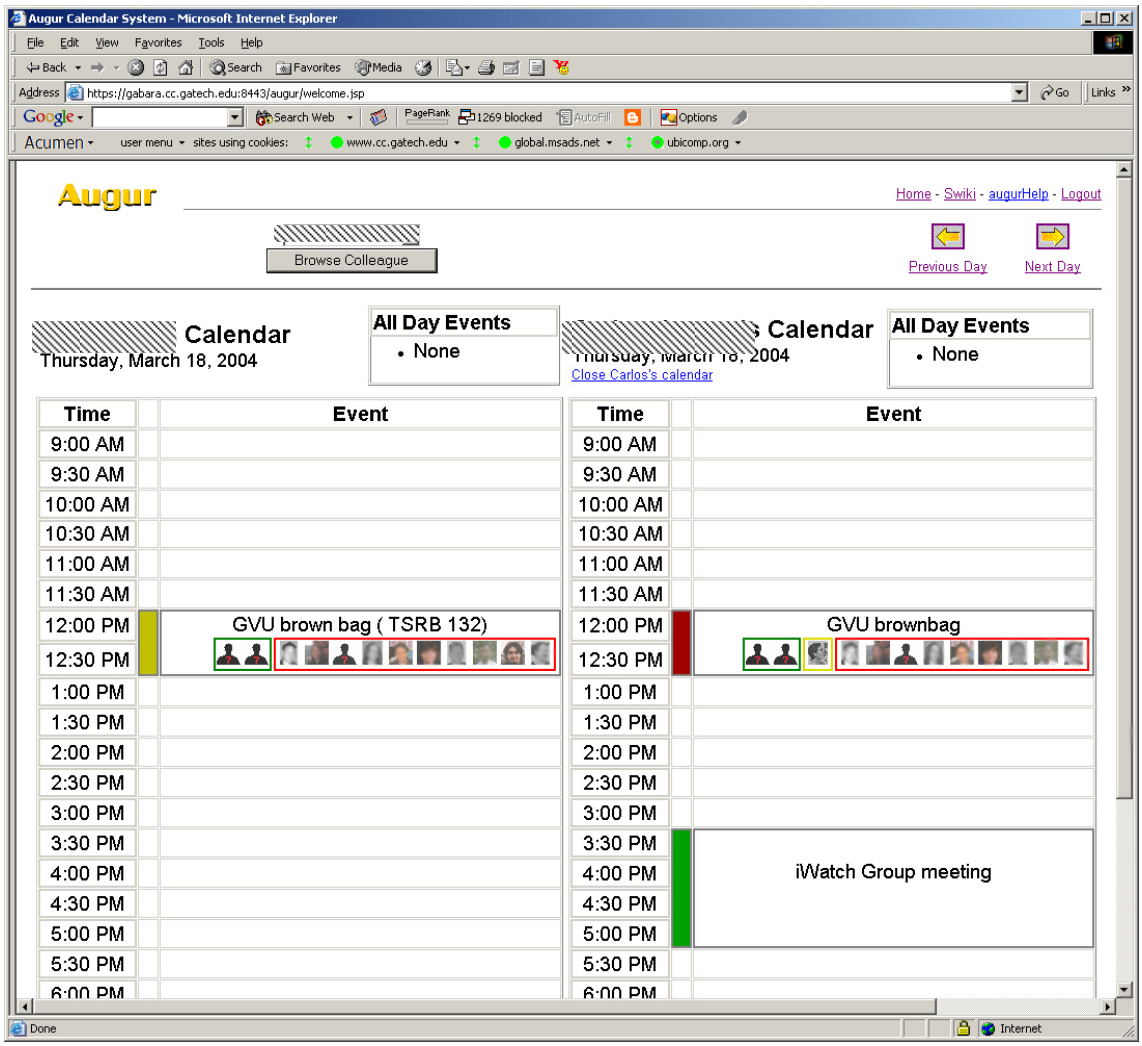


Figure 46: Augur Interface Example 2

To obtain calendar information, Augur contains software that allows it to use existing calendar data from other calendar systems. For example, Augur contains PalmOS conduit software that automatically sends calendar information to our parsing module when a PalmOS PDA is synchronized with a networked computer. For iCal and Mozilla Thunderbird users, we have web-based tools that will automatically update Augur upon changes to calendar data. The parsing module reads the formats of these external calendars and updates a table of events in the central database with this information.

Once the latest calendar data is retrieved, our prediction and event matching modules insert additional information into the database. The prediction module uses a Bayesian network to add information about the likelihood of attendance for future events. Each user is associated with a separate copy of the network that is capable of learning their individual attendance habits over time. An additional component allows users to provide examples to the system by submitting daily attendance checklists via the web. The event-matching module uses text-processing techniques to identify events from other colleagues' calendars that are likely to represent the same event.

With current, augmented calendar data now present in the database, web-based visualizations display this information to users through the user interface. The daily calendar view, described previously, displays a user’s scheduled events along with information about whom he/she might see at those events. Additional software logs accesses to the visualizations and stores this information in the database, but is currently only used for the purposes of scientific study.

APPENDIX B: PARTICIPANT SURVEY

Background Survey:

Please answer the following questions before doing the analysis, but after reading the papers and instructions. These questions will help us judge your general CS background and experience.

What is your academic major?

How many years have you been in college?

List all 3000 and 4000 level CS-classes taken at tech:

Which best describes your experience with the following (circle best fit)

Goal-decomposition:

Never heard of it, Some familiarity, Have used it, Use it regularly, Expert

Heuristic evaluation:

Never heard of it, Some familiarity, Have used it, Use it regularly, Expert

Use-cases:

Never heard of it, Some familiarity, Have used it, Use it regularly, Expert

Object-sequence diagrams:

Never heard of it, Some familiarity, Have used it, Use it regularly, Expert

Other relevant design/analysis techniques (Please list name & level of expertise):

How familiar are you with Teamspace or similar meeting/class/project capture, access, and review systems

Never heard of it, Some familiarity, Have used it, Use it regularly, Expert

If you have used such systems, please list their names:

Time estimate:

As you do the analysis, please answer track the time spent on the different activities here. Please do this as accurately as possible. These estimates will not be associated with a grade so please be honest. I ask that you please spend 2-3 hours on this analysis not counting time spent reading the assigned papers. Overall, you should expect to set aside 3-4 hours for this experiment.

Give a breakdown by activity here:

Reading papers:

[STRAP]

Doing goal-decomposition:

Applying identification questions:

Applying heuristics:

[/STRAP]

[Patrick & Kenny]

Deriving Use cases:

Drawing Object-sequence diagrams:

Applying usability principles:

[/Patrick & Kenny]

[Hong et al]

Identifying problems

Asking social and organizational questions:

Asking technology oriented questions:

Analysis of problems:

[/Hong et al]

Write-up

Problem report

After completing your analysis, please list all problems discovered here. Give enough context and details so we know what you mean by each, and what causes the problem. If you need more space, add an extra page.

If you were asked to draw diagrams or do other writing as part of your method, report these separately (listing the problems you discovered there as well).

Debriefing

After completing your analysis, please answer the following questions about your experience:

The system (Teamspace) I was asked to analyze was well documented

1	2	3	4	5
Strongly Disagree				Strongly Agree

The analysis method I was asked to use was well documented

1	2	3	4	5
Strongly Disagree				Strongly Agree

The method assigned was well thought out (made sense)

1	2	3	4	5
Strongly Disagree				Strongly Agree

The method assigned was easy to learn

1	2	3	4	5
Strongly Disagree				Strongly Agree

The method assigned was easy to use

1	2	3	4	5
Strongly Disagree				Strongly Agree

The results from this analysis would be useful in a real project

1	2	3	4	5
Strongly Disagree				Strongly Agree

This analysis would be worthwhile performing in a real project

1	2	3	4	5
Strongly Disagree				Strongly Agree

Which parts of the method described (if any) were confusing to you (list):

Which parts of the method described (if any) were most helpful to you (list):

Which parts of the method described (if any) were least helpful to you (list):

Any other feedback and opinions:

APPENDIX C: TEAMSPACE

Excerpt from:
Richter, H., Abowd, G., Geyer, W., Fuchs, L., Daijavad, S., Poltrock, S. (2001) "Integrating Meeting Capture within a Collaborative Team Environment", In Proceedings of Ubicomp 2001, ACM Conference on Ubiquitous Computing, Atlanta, GA, USA.

TeamSpace is implemented as a mostly web-based application. This allows it to be accessible from a large number of platforms with no installation. Meeting activities can be thought of in three phases: preparation before the meeting, conducting of the meeting, and later review of the meeting. Each of these phases mainly corresponds to one piece of the TeamSpace prototype.

1 Meeting Preparation: Main TeamSpace Interface

Bill, the team lead, prepares for a weekly status meeting planned the next day. He checks the meeting information in TeamSpace to make sure the rooms are scheduled and adds a few guest participants. He checks which action items generated discussion last week, and adds those to the meeting. Finally, he adds a small presentation he has prepared for the meeting. The participants are then automatically emailed a meeting invitation.

Meeting preparation is accomplished using the main TeamSpace interface. Figure 2 shows a screenshot of this interface. After logging in, users are taken to their starting page which highlights the current day's meetings and open action items. Users can access additional information using the context tabs of *People*, *Meeting*, and *Task*. The *Meeting* tab provides both a calendar and list view of meetings. The list view provides mechanisms to filter and search the meeting list. Under *Task* a user can view her own action items or browse and search the entire team list. The lower half of the window is a document view for displaying and editing the details of each individual object. For the preparation scenario above, Bill would go to the *Meeting* tab to create a new meeting, then enter in all of the information in the document view. Invited participants would then see that meeting on their calendar or meeting list when they logged into TeamSpace.

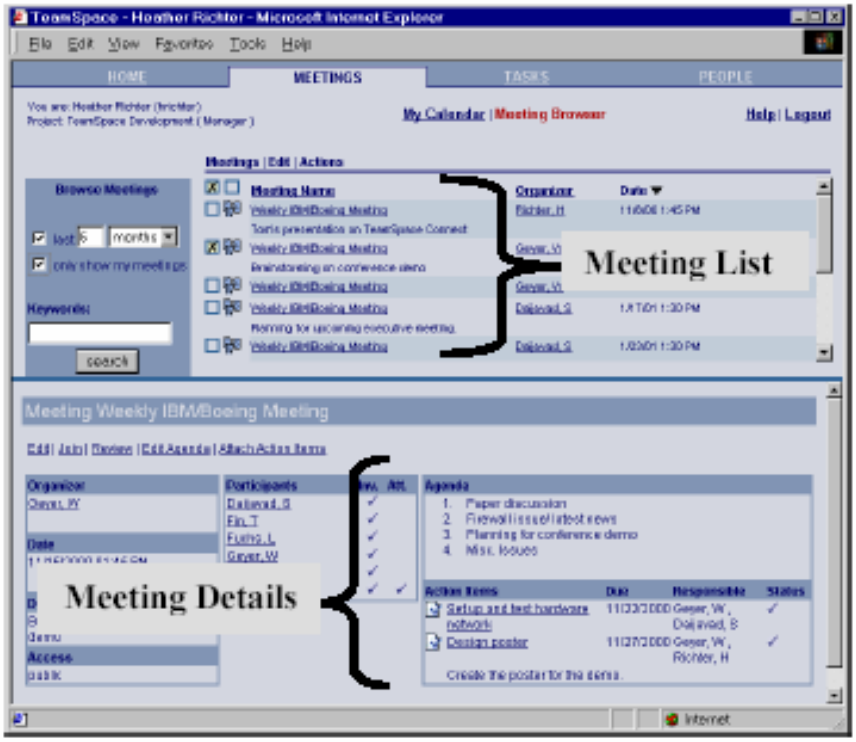


Figure 47: Screenshot of The Main Teamspace Interface

The user is viewing the details of one captured meeting. The top is the meeting list view, the bottom pane is the detailed view.

Besides the meeting environment, TeamSpace is intended to support other activities such as project management, document management, and team awareness and communication. In other words, this interface is meant to serve as the main portal for all team activities, including meeting capture and access.

2 Meeting Capture: MeetingClient

The day of the meeting, team members in Seattle gather in their conference room. Mary, the meeting facilitator arrives a few minutes early to log into TeamSpace and start the meeting. Meanwhile, team members in other locations enter the virtual meeting from their desktop browsers. After team members greet one another and chat for a few minutes, Mary opens the meeting agenda. She adds any new items proposed by the team.

Pushing the agenda aside, she opens the action item list to get an update on each of the unfinished tasks. As each team member lists their progress, Mary updates the item list, marking off items, changing items, and adding new action items.

The next item on the agenda is a presentation by Bill about an interface problem just discovered. Bill opens the presentation and explains the problem. Working at her desktop in St. Louis, Sally circles a region on one of the components and notes some of the manufacturing constraints that influenced its design. Also from his desktop, Jim draws a sketch to explain the reason for these constraints. The presentation spawns a brainstorming session for solutions. The team sketches their ideas on the whiteboard, with distributed team members drawing at their desktops.

The meeting capture phase is supported through the MeetingClient interface, shown in Figure 3. MeetingClient is launched automatically on a client's machine when joining a meeting. This client provides viewing, editing, and annotating of agendas and action items, as well as viewing and annotating of PowerPoint presentations. Thus, MeetingClient records events such as joining and leaving a meeting; viewing, editing, and checking off agenda items; viewing, editing, and creating action items, and viewing and annotating presentations. Participants are not required to use or interact with any of these objects.

The panel on the left of Figure 3 provides an overview and navigation of the meeting. The list of agenda items, action items, presentations and invited participants can be seen and individual items can be selected. The main view shows the selected presentation, or the agenda or action item editor. The toolbar at the bottom of the screen contains the pen and text tools. In the above scenario, Mary would begin the meeting by selecting "Agenda" in the overview panel. She would then edit and rearrange the agenda.

Next, she would move to the action item view to go through the list of action items. Bill would then view his presentation, which everyone could annotate. Finally, Bill would create a blank presentation to function as a whiteboard for the brainstorming session.

Additionally, MeetingClient provides low-bandwidth video, which is viewed in a separate window, providing real-time awareness of other team members. All of the meeting data and events remain synchronized between clients, and are automatically time-stamped and stored on the server. MeetingClient does not impose any floor control for the meeting, thus leaving the potential for conflict and unpredictable results.

Figure 48: Screenshot of MeetingClient

The user is viewing an annotated presentation. The overview bar on the left of the screen shows the agenda, action items, presentations, and participants.

The proposed architecture is simple. The Server stores all data permanently; audio, video, and event streams as raw data on the server's file system, all other information, such as action items and meeting descriptions, are stored in a database. The client connects directly to the server.

3 Meeting Access: MeetingViewer

During the meeting, the team was split between two different solutions to their problem and needed more discussion to make a decision. Pat and Jim meet later that day to further discuss one of the proposed solutions. During their meeting, which they also capture, they recall a contact Sally mentioned during the earlier meeting. They open up the meeting and browse through it by using a timeline that indicates the spots where Sally was talking.

Chris had to leave the weekly status meeting early. Before he tackles his new tasks, he returns to the meeting records to listen to the portions he missed. He skims the meeting with a time slider by jumping from one agenda item to the next. Then he dives into Bill's presentation, using a thumbnail navigation, to replay the portion of the presentation where the group talked about the manufacturing constraints. Chris also listens to the comments Dave made to Pat and Jim during their conversation.

One year later, Bill is leading a team that runs into a similar component interface problem. He asks Harry, one of his team members, to look at the problem the old project had, and why they chose their solution. Harry reads the documentation, accesses the meeting records and replays pieces of various discussions, and prepares a presentation using some of the older material to present his findings.

After a meeting is completed, the meeting records are automatically available. Users can select completed meetings in TeamSpace and launch a MeetingViewer applet to view and playback these meetings.

The MeetingViewer, shown in Figure 5, integrates all of the meeting information based on time. The viewer uses a two-scale timeline for navigating a set of selected

meetings, providing random access playback. The timeline is painted with interesting events as both a visual summary of the meeting, and as an aid for navigation.

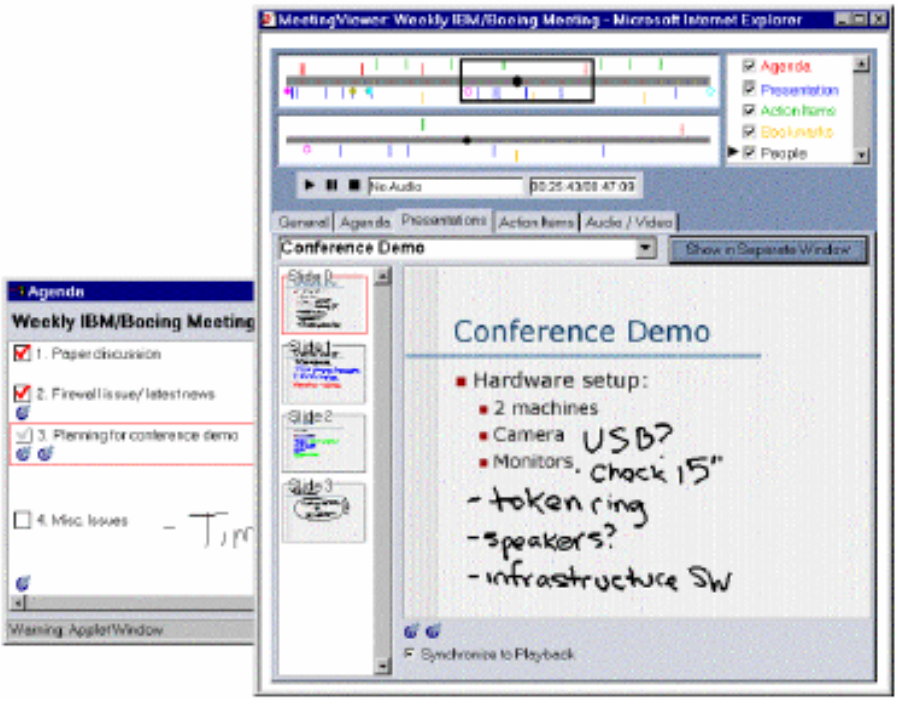


Figure 49: Screenshot of MeetingViewer Showing a Single Meeting
MeetingViewer can also be used to view multiple meetings.

Interesting events currently are people joining and leaving, agenda items being discussed, action items visited or created, and slides visited, but could include any envisioned events such as people speaking and keyword locations. Users can control which of these events they view and can use the events to find relevant portions within a meeting to playback. Playback of a meeting not only involves playing the audio and video, but also involves playback of all of the recorded events of a meeting such as slide visits or agenda item discussion.

The remainder of the meeting information is displayed on a series of tabbed panes for each of the objects related to the meeting, including descriptions and summaries of the meeting, agenda, presentations, action items, and video images. These

panes are a very general approach for displaying a large amount of related information. However, to enable customized views, each pane can be opened in a separate window, moved and resized. In this way, users can view any subset of the information they wish at once. Additionally, as we add more objects to TeamSpace, we can easily add more meeting-related objects to this interface as another tabbed pane, such as documents that were reviewed or referenced during the meeting.

REFERENCES

Abowd, G. and Mynatt, E. D. (2000) "Charting past, present, and future research in ubiquitous computing." *ACM Transactions on Computer-Human Interaction, 7(1):29-58*.

Ackerman, M. and Cranor, L. (1999) "Privacy Critics: UI Components to Safeguard Users' Privacy." *Extended abstracts of ACM Conference on Human Factors in Computing Systems CHI'99*, 2, 258-259.

Adams, A. and Sasse, A. (2001) "Privacy in Multimedia Communications: Protecting Users, Not Just Data". In A. Blandford, J. Vanderdonkt & P. Gray (Eds.): *People and Computers XV - Interaction without frontiers. Joint Proceedings of HCI2001 and ICM2001*, Lille, Sept. 2001. pp. 49-64. Springer.

Adkinson, W. F., Eisenach, J. A., and Lenard T. M. (2002) "Privacy Online: A Report on the Information Practices and Policies of Commercial Web Sites" Progress and Freedom Foundation, Washington DC. March 2002

Anderson, R. J. (2001) *Security Engineering: A Guide to Building Dependable Distributed Systems*, Wiley Computer Publishing

Antón, A.I., Earp, J.B., Bolchini, D., He, Q., Jensen, C., and Stufflebeam, W. (2004) "The Lack of Clarity in Financial Privacy Policies and the Need for Standardization", *IEEE Security & Privacy*, 2(2), pp. 36-45, 2004.

Antón, A. I., Earp, J. B. and Reese, A. (2002) "Analyzing Web Site Privacy Requirements Using a Privacy Goal Taxonomy", *IEEE Requirements Engineering Conference (RE'02)*, Essen, Germany, September 2002.

Antón, A.I., and Potts, C. (1998) "The Use of Goals to Surface Requirements for Evolving Systems." *International Conference on Software Engineering (ICSE `98)*, Kyoto, Japan, pp. 157-166, 19-25 April 1998

Antón, A.I. (1996) "Goal-Based Requirements Analysis" Second IEEE International Conference on Requirements Engineering (ICRE `96) , Colorado Springs, Colorado, pp. 136-144, 15-18 April 1996

Ashley, P. and Schunter, M. (2002) "The Platform for Enterprise Privacy Practices". *Information Security Solutions Europe*, Paris France, October 2002.

Bellotti, V. (1997) "Design for Privacy in Multimedia Computing and Communications Environments." In P. Agre, & M. Rotenberg (eds.). *Technology and Privacy: The New Landscape*. MIT Press: Cambridge.

Bellotti, V. and Sellen, A. (1993) "Design for Privacy in Ubiquitous Computing Environments". *Proceedings of 3rd European Conf. on Computer Supported Cooperative Work, (ECSCW 93)*, 77-92

Black, E., *IBM and the Holocaust: The Strategic Alliance between Nazi Germany and America's Most Powerful Corporation*. Crown Publishers, NY, NY: (2001)

Bly, S., Harrison, S., and Irwin, S. (1993). "Media Spaces: Bring People Together in Video, Audio and Computing Environments." *Communications of the ACM*, 36(1), 28-47.

Bohrer, K, Levy, S., Liu, X., and Schonberg, E.. (2003) "Individualized Privacy Policy Based Access Control" *Proceedings 6th International Conference on Electronic Commerce Research (ICECR-6)*, October 2003, Dallas, Texas, USA

Boehm, B.W. (1985) "A Spiral Model Of Software Development And Enhancement," *2nd. International Software Process Workshop*. Coto de Caza, Trabuco Canyon, USA 1985. Wileden, J. and Dowson, M. (Eds.)

Boehm, B.W. (1981) *Software Engineering Economics.* NY: Prentice Hall.

Bruckman, A., and Resnick. M. (1995) "The MediaMOO Project: Constructionism and Professional Community." *Convergence* 1:1, pp 94-109, Spring 1995.

Butler S. A. and Fischbeck P. (2002) "Multi-Attribute Risk Assessment" *Proceedings of SREIS'02*, Raleigh, NC.

Camp, L.J. (1999) "Web Security And Privacy: An American Perspective." *Information Society*, 0197-2243, October 1, 1999, Vol. 15, Issue 4

Card, S., Morn, Thomas P., and Newell, A., (1980) "The keystroke-level model for user performance with interactive systems", *Communications of the ACM, 23* (1980), 396-210

Cheng, L., Franham, S., and Stone, L. (2002). "Lessons Learned: Building and Deploying Shared Virtual Environments," In Schroeder, R (ed.), *The Social Life of Avatars: Presence and Interaction in Shared Virtual Environments.* London: Springer-Verlag. 90-111.

Cranor L. (2002) *Web Privacy with P3P*. O'Reilly & Associates, September 2002.

Cranor, L., Arjula, M., and Guduru, P. (2002) "Use of a P3P User Agent by Early Adopters." *Proceedings of the ACM Workshop on Privacy in the Electronic Society*, November 21, 2002, Washington, DC.

Computer Research Association (CRA) (2003) Conference on "Grand Research Challenges in Information Security and Assurance." http://www.cra.org/Activities/grand.challenges/security/. November 16-19, 2003.

Culnan, M. J. and Milne, G. R. (2001) *The Culnan-Milne Survey on Consumers & Online Privacy Notices: Summary of Responses*. Washington DC: FTC, December 2001.

Culnan, M.J. (1999) *Georgetown Internet Privacy Policy Survey: Report to the Federal Trade Commission*. Washington, DC: Georgetown University, McDonough School of Business.

Dardenne, A., Lamsweerde, A.V. and Fickas, S. (1993) "Goal-directed requirements acquisition". *Science of Computer Programming*, 20:3--50.

Denning. D.E. (1993) "The Clipper encryption system." *American Scientist*, 81(4):319

Dourish, P. (1993) "Culture and Control in a Media Space" *Proceedings of the Third European Conference on Computer-Supported Cooperative Work 1993* p.125-137

Earp, J.B, Antón, A.I, Aiman-Smith, L and Stufflebeam, W.H. (2005) "Examining Internet Privacy Policies Within the Context of User Privacy Values" in *IEEE Transactions on Engineering Management,* Vol. 52., No. 2, May 2005.

Earp J.B. and Meyer, G. (2000) “Internet Consumer Behavior: Privacy and its Impact on Internet Policy”, *28th Telecommunications Policy Research Conference*, Sept. 23-25, 2000.

Erickson, T. and Kellogg, W.A. (2000). “Social translucence: An approach to designing systems that mesh with social processes.” *ACM Transactions on Computer-Human Interaction*, Vol. 7, No. 1, pp. 59-83.

European Union (EU). (1995). *Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data.* http://europa.eu.int/eur-ex/en/lif/dat/1995/en_395L0046.html

Federal Trade Commission (FTC) (2000) *Privacy Online: Fair Information Practices in the Electronic Marketplace. A Report to Congress.*

Flesch, M. (1949) *The Art of Readable Writing*, Macmillan Publishing, 1949

Foucault, M. (1983) "The Subject and Power" Afterword. *Michel Foucault: Beyond Structuralism and Hermeneutics*. 2nd ed. Hubert Dreyfus and Paul Rainbow. Chicago: University of Chicago Press, 1983. 208-26.

Garfinkel, S. (1995) *PGP: Pretty Good Privacy*. O’Reilly and Associates.

Goecks, J. and Mynatt, E.D. (2005) “Leveraging Social Networks for Information Sharing”. In *Proceedings of CSCW 2004*, p. 328-331.

Goecks J. and Mynatt, E.D. (2005). “Supporting Privacy Management via Community Experience and Expertise” To appear in *Proceedings of 2005 Conference on Communities and Technology*.

Goldberg, I., Wagner, D., and Brewer, E. (1997) “Privacy-enhancing Technologies for the Internet.”. *Proceedings of IEEE COMPCON 97.*

Grudin, J. (2004) “Managerial Use and Emerging Norms: Effects of Activity Patterns on Software Design and Deployment”. *Proceedings of Hawaii International Conference on System Science 37*, 2004.

Grudin, J. (2001) “Desituating Action: Digital Representation of Context”. *Human-Computer Interaction, 16, 2-4*

Grudin, J. and Palen, L. (1997) “Emerging Groupware Successes in Major Corporations: Studies of Adoption and Adaptation”. *International Conference on Worldwide Computing and Applications*, 142-153.

Grudin, J. (1994) “Groupware and social dynamics: Eight challenges for developers”. *Communications of the ACM, 37, 1.*

Habermas, J. (1989) *The Structural Transformation of the Public Sphere: An Inquiry into a Category of Bourgeois Society*. Trans. Thomas Burger. 1962; Cambridge: MIT Press, 1989.

Harris Interactive (1998) *E-commerce & privacy: What net users want.* Privacy and American Business & PriceWaterhouseCooper LLP. Online: http://www.pandab.org/ecommercesurvey.html (accessed Feb 12, 2005).

Harris Interactive (2001) *Privacy Notices Research Final Results*, conducted for the Privacy Leadership Initiative, Rochester, NY, 2001. Online: http://www.ftc.gov/bcp/workshops/glb/supporting/harris%20results.pdf

Harris Interactive (2003) *The Harris Poll® #17: Most people are ‘privacy pragmatists’ who, while concerned about privacy, will sometimes trade it off for other benefits.* Online: http://www.harrisinteractive.com/harris_poll/index.asp?PID=365 (accessed Feb 12, 2005)

Hartson, H. R. and Hix, D. (1989) “Toward empirically derived methodologies and tools for human computer interface development.” *Int. J. Man-Machine Studies*, 31, 477-494.

Hong, J.I., Ng, J., Lederer, S., and Landay, J.A.. (2004) “Privacy Risk Models for Designing Privacy-Sensitive Ubiquitous Computing Systems”. *Designing Interactive Systems (DIS2004)*. Boston, MA.

Horn, S. (1998) *Cyberville: Clicks, culture, and the creation of an online town*. New York: Warner Books

Horvitz, E. (1999) “Principles of Mixed-Initiative User Interfaces”. *Proceedings of CHI'99*, May 1999, pp.159-166.

Isaacs, E., Walendowski, A., Whittaker, S., Schiano, D.J. and Kamm, C., (2002) “The Character, Functions, and Styles of Instant Messaging in the Workplace.” *Proceedings of ACM Conference on Computer Supported Cooperative Work (CSCW 2002)*, (New Orleans, LA, 2002), New York, NY: ACM Press.

Javelin Strategy & Research (2005), *2005 Identity Fraud Survey Report*, January 2005, Online: http://www.javelinstrategy.com/reports/2005IdentityFraudSurveyReport.html

Jensen C., Potts C., Jensen C.. (2005) “Privacy Practices of Internet Users: Self-report versus Observed Behavior” *International Journal of Human Computer Studies*. July 2005.

Jensen. C. and Potts, C. (2004) “Privacy Policies as Decision-Making Tools: A Usability Evaluation of Online Privacy Notices” *Proceedings of CHI’04* Vienna, Austria, April 2004

John, B.E. and Kieras, D.E. (1996) “The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast”. *ACM Transactions on Computer-Human Interaction, 3 (4).*

Jupiter Research (2002) “Security and Privacy Data.” *Presentation to the FTC Security Workshop*, May 20, 2002

Kobsa, A. (2002). “Personalized hypermedia and international privacy.” *Communications of the ACM*, 45(5), 64-67.

Langheinrich, M. (2001) “Privacy by Design - Principles of Privacy-Aware Ubiquitous Systems.” *Proceedings of Ubicomp 2001*, pp. 273-291, Springer-Verlag LNCS 2201

Lessig, L. (1999) *Code and Other Laws of Cyberspace.* Basic Books, New York, 1999.

Long, W. J., and Pang Quek, M. (2002). “Personal data privacy protection in an age of globalization: the USEU safe harbor compromise.” *Journal of European Public Policy*, 9 (June), 325-344.

Mayhew, D.J., (1999). *The usability engineering lifecycle*. San Francisco CA USA: Morgan Kaufman.

National Institute of Standards and Technology (U.S.) (1994) *Digital Signature Standard* Federal Information Processing Standards Publication 186, May 1994.

National Telecommunications and Information Administration (U.S.) (2002). *A Nation Online: How Americans Are Expanding Their Use of the Internet.* Washington, D.C. February 2002

Nielsen, J. & Molich, R. (1990). "Heuristic evaluation of user interfaces", *Proceedings of CHI'90*. Seattle, WA, 1-5 April, 249-256.

Nielsen, J., and Landauer, T. K. (1993). "A mathematical model of the finding of usability problems." Proceedings of ACM/IFIP INTERCHI'93 Conference. Amsterdam, The Netherlands, April 24-29, 206-213.

Nielsen, J. (1994). "Heuristic evaluation." In Nielsen, J., and Mack, R.L. (Eds.), *Usability Inspection Methods*, John Wiley & Sons, New York, NY.

Norman, D. A. Psychology of Everyday Things. Basic Books, 1988

Organisation of Economic Cooperation and Development (OECD). (1980). *Guidelines on the Protection of Privacy and Transborder Flows of Personal Data* .http://www.oecd.org/document/18/0,2340,en_2649_34255_1815186_1_1_1_1,00.html

Orwell, G. (1949). *Nineteen Eighty-Four*. London: Martin Secker & Warburg.

Palen, L. and Dourish, P. (2003) "Unpacking 'Privacy' for a Networked World." *Proceedings of CHI'03*, Ft. Lauderdale, FL. 2003

Palen, L. and Grudin, J. (2002) "Discretionary Adoption of Group Support Software: Lessons from Calendar Applications". *Implementing Collaboration Technologies in Industry*. B. E. Munkvold, Springer Verlag.2002

Palen, L. (1999) "Social, Individual, and Technological Issues for Groupware Calendar Systems". *Proceedings of CHI '99*, Pittsburgh, PA, pp. 17-24.

Patrick, A.S., & Kenny, S. (2003). “From Privacy Legislation to Interface Design: Implementing Information Privacy in Human- Computer Interfaces.” In R. Dingledine (Ed.), *Proceedings of Privacy Enhancing Technologies Workshop (PET2003)*, Dresden, Germany, 26-28 March, 2003. LNCS 2760, pp. 107-124.

Potts, C. (1999) “ScenIC: A Strategy for Inquiry-Driven Requirements Determination”, *IEEE Fourth International Symposium on Requirements Engineering (RE’99)*, University of Limerick, Ireland, pp. 58-65, 7-11 June 1999.

Potts C., and Catledge C. (1996) “Collaborative Conceptual Design: A Large Software Project Case Study”. *Computer-Supported Cooperative Work: The Journal of Collaborative Computing*, *5*:.414-445

Preece, J., Rogers, Y., & Sharp, H. (2002). *Interaction Design: Beyond Human-Computer Interaction*. New York, NY: John Wiley & Sons.

Purcell, A. and Gero, J. (1996) “Design and Other Types of Fixation.” *Design Studies*, 17, 1996, pp 363-383.

Rand, A. (1943) *The Fountainhead*, Signet Book: New York.

Rhodes, B.J. (2000) “Margin notes: building a contextually aware associative memory”. *Proceedings 5th International Conference on Intelligent User Interfaces (IUI'2000).*

Richter, H., Abowd, G., Geyer, W., Fuchs, L., Daijavad, S., Poltrock, S. (2001) "Integrating Meeting Capture within a Collaborative Team Environment", *In Proceedings of Ubicomp 2001, ACM Conference on Ubiquitous Computing*, Atlanta, GA, USA.

Royce, W.W. (1970) “Managing the Development of Large-Scale Software: Concepts and Techniques” *Proceedings of Wescon*, August 1970

Rumbaugh, J.E., Blaha, M.R., Premerlani, W.J., Eddy, F., Lorensen, W.E. (1991) *Object-Oriented Modeling and Design* Prentice-Hall 1991.

Shepherd, A. (1985). “Hierarchical task analysis and training decisions.” *Programmed Learning and Educational Technology*, 22, 162-176.

Smith, I., LaMarca, A., Consolvo, S., and Dourish, P. (2004) “A social approach to privacy in location-enhanced computing.” *In Proceedings of the Workshop on Security and Privacy in Pervasive Computing, 2004*.

Synovate (2003) *Identity Theft Survey Report*, prepared for the Federal Trade Commission (FTC). September 2003. Online: http://www.ftc.gov/os/2003/09/synovatereport.pdf

Tullio, J., Goecks, J., Mynatt, E.D., and Nguyen, D.H. (2002) “Augmenting Shared Personal Calendars”, *Proceedings of UIST 2002*, 11-20.

United States (US) (1998) *Children’s Online Privacy Protection Act of 1998*, Public Law No. 105-277, October 21, 1998.

United States (US) (1999) *Gramm-Leach-Bliley Financial Modernization Act of 1999*, Public Law No. 106-102, November 1, 1999.

United States (US) (1996) *Health Insurance Portability and Accountability Act of 1996*, Public Law No. 104-191, August 21, 1996.

United States Department of Health, Education and Welfare (HEW) (1973) *The Code of Fair Information Practices.*

Van Lamsweerde, A. (2004) “Elaborating Security Requirements by Construction of Intentional Anti-Models”. *Proceedings of ICSE 2004*: 148-157

Warren, S. and Brandeis, L. (1890) “The right to privacy.” *Harvard Law Review*, 4:193 – 220, 1890.

Weirich D. and Sasse, M.A. (2001) “Pretty Good Persuasion: A first step towards effective password security for the Real World.” *Proceedings of the New Security Paradigms Workshop 2001* (Sept. 10-13, Cloudcroft, NM), pp. 137-143. ACM Press.

Westin, A.F. (1967). *Privacy And Freedom*. New York: Atheneum.

Whitten, A. and Tygar, J.D. (1999) "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0," *Proceedings of the 8th USENIX Security Symposium.*

Yu, E., and Cysneiros, L. (2002) “Designing for Privacy and Other Competing Requirements” *2nd Symposium on Requirements Engineering for Information Security (SREIS’02)*. Raleigh, North Carolina, October 16, 2002.

Yu, E. (1997) “Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering” *in Proceedings Of the 3rd IEEE International Symposium on Requirements Engineering*, pp:226-235, 1997.

VITA

CARLOS JENSEN

JENSEN was born in Madrid, Spain. He attended public schools in Rælingen, Norway, before starting his undergraduate education in Informatics at the University Of Oslo, Norway before transferring and earning a B.S. in computer Science in 1998 from the State University of New York, College at Brockport, New York. In 1998 he came to Georgia Tech to pursue a doctorate in Computer Science, with an emphasis on Human-Computer Interaction. When he is not working on his research, Mr. Jensen enjoys spending time with his wife Olga and daughter Anna Maria. Mr. Jensen is also an avid supporter of the Olympic movement and has attended (as an observer) the last three Olympic Games.